AFRL-RY-WP-TR-2009-1031

ROBUST DATA ALIGNMENT AND ITS APPLICATION IN AN INTEGRATED MULTI-SENSOR SURVEILLANCE SYSTEM CONSISTING OF UAVs AND FIXED PLATFORMS

The Current and Future Phase of the RASER Project

Ü. Özgüner, S. Jwa, J. Martin, and K. Redmill

The Ohio State University

JANUARY 2009
Final Report

Approved for public release; distribution unlimited.

See additional restrictions described on inside pages

STINFO COPY

AIR FORCE RESEARCH LABORATORY
SENSORS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7320
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the USAF 88th Air Base Wing (88 ABW) Public Affairs Office (PAO) and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RY-WP-TR-2009-1031 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

*//Signature//

OLGA MENDOZA-SCHROCK
Research Mathematician
ATR & Fusion Algorithms Branch
Sensor ATR Technology Division

//Signature//

CHRISTINA G. SCHUTTE, Chief
ATR & Fusion Algorithms Branch
Sensor ATR Technology Division

//Signature//

STEVEN P. WEBBER, Lt Col, USAF
Deputy, Sensor ATR Technology Division
Sensors Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

*Disseminated copies will show "//Signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS**.

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
January 2009	Final	10 March 2005 – 31 January 2009

4. TITLE AND SUBTITLE
ROBUST DATA ALIGNMENT AND ITS APPLICATION IN AN INTEGRATED MULTI-SENSOR SURVEILLANCE SYSTEM CONSISTING OF UAVs AND FIXED PLATFORMS
The Current and Future Phase of the RASER Project

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8650-05-1-1854

5c. PROGRAM ELEMENT NUMBER
62204F

6. AUTHOR(S)
Ü. Özgüner, S. Jwa, J. Martin, and K. Redmill

5d. PROJECT NUMBER
6095

5e. TASK NUMBER
04

5f. WORK UNIT NUMBER
60950419

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Ohio State University
2015 Neil Avenue
Columbus, OH 43210

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
Sensors Directorate
Wright-Patterson Air Force Base, OH 45433-7320
Air Force Materiel Command
United States Air Force

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/RYAT

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-RY-WP-TR-2009-1031

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES
PAO Case Number: 88ABW-09-0772; Clearance Date: 26 Feb 2009. This report contains color.

14. ABSTRACT
This report introduces a new algorithm called Robust Data Alignment (RDA) for persistent sensing in multi-layered sensor network. Invariant features collected from a set of images are used for image registration. Information-theoretic cost models a functional structure where an optimal transformation can be recovered from feature representations of different but related images. By using a cooperative search strategy, we believe that RDA can help to understand and deal with many situations in persistent sensor networks by solving problems such as Layered sensing, Multi-modal data fusion, Multi-UAV sensing. In addition, we provide Georegistration for CLIF 2007 dataset. SIFT algorithm after georeferencing recovers a correspondence for data registration.

15. SUBJECT TERMS
data registration, persistent sensing, layered data fusion, geo-referencing, information-theoretic alignment, mini-UAV sensing, multi-modal data fusion, multi-layered sensor networks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Olga Mendoza-Schrock
Unclassified	Unclassified	Unclassified	SAR	96	19b. TELEPHONE NUMBER *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Table of Contents

Section **Page**

1. Executive Summary 1
2. Introduction 2
3. Information-theoretic Data registration 4
3.1 Introduction 4
3.2 Problem Formulation 6
3.2.1 Data Alignment Problem 6
3.3 Cost and Search Strategy 7
3.3.1 Cost Criteria 8
3.3.2 Search Strategies 10
3.4 Experimental Results 11
3.4.1 RDA vs. Correspondence-based Method 11
3.4.2 Mosaic Image from a Video Sequence 12
3.4.3 Monte Carlo Simulation 16
3.4.4 Data Refinement 17
4. Layered Data Fusion: Multi-UAV Sensing over Urban Areas 20
4.1 Introduction 20
4.2 Data alignment in Multi-layered Sensor Networks 22
4.3 Scaling Factor Analysis 23
4.3.1 Scaling Factor Analysis 23
4.3.2 Weighted Feature Points 25
4.4 Experimental Results 25
4.4.1 Video Registration 25
4.4.2 Simultaneous Registration and Vehicle Detection 26
5. Problems for Persistent Sensing 28
5.1 Introduction 28
5.2 Three problems for Persistent Sensing 29
5.2.1 Scale Space Theory: Review 30
5.3 Methodology: Extended RDA 32
5.3.1 Projective Registration 32
5.3.2 Gaussian Scale-space Based Estimation of a Suboptimal Feature Set for RDA 34
5.4 Experimental Results 36
5.4.1 Multi-level Registration 36
5.4.2 Multimodal Data Registration 37
5.4.3 Mosaic Construction and Scene Localization 39
6. Georegistration based on Georeferencing 42
6.1 Introduction 42
6.2 Georeferencing Images 42
6.2.1 Longitude, Latitude to Pixel Locations 43
6.2.2 CLIF 2007 Camera 0 Orientation 44
6.2.3 Perspective Projection 45
6.2.4 Lens Distortion 45

Section **Page**

6.2.5 Finding the Camera Parameters 46
6.3 Image to Image Registration 51
6.3.1 The Affine Transform 51
6.3.2 Using Correspondences to find an Affine Transform 51
6.3.3 RANSAC Preprocessing 52
6.4 Finding Correspondences 53
6.4.1 Point Pattern Matching 53
6.4.2 Aerial Imagery 54
6.4.3 Error in the Non-Georeferenced Example 55
6.4.4 Error with the Georeferenced CLIF2007 Images 100 110 56
6.4.5 SIFT 57
6.4.6 Two-Image SIFT Example 58
6.4.7 CLIF2007 and Electro-Optical Mid-Level Images 59
6.5 Georeferencing the CLIF 2006 Data 60
6.5.1 Errors from IMU, Calibration in CLIF 2006 62
6.5.2 Errors from IMU, Calibration in CLIF 2007 64
6.5.3 Post SIFT Feature Stabilization Error 65
6.5.4 Problems Encountered with the CLIF2006 Data 66
6.6 Conclusion with Recommendations for Future Use of SIFT 67
7. Conclusion 68
Appendix A 69
General Description of How to Use RDA Code 69
Appendix B 73
Code (“stabilize_one_frame”, “geo_ref”) Use Example 73
8. References 81
LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS 83

List of Figures

Figure **Page**

1. Integrated Process: target recognition, tracking, and monitoring tasks can utilize the geographic or reference map generated from data registration 5
2. Two Cases of Varying Error-bounds in RDA 9
3. Two data sets are marked as red (+) marks and registered outputs are shown; without noise both RDA and LS give almost the identical results 12
4. Robustness of the New Approach: (a) With noise we compare RDA and LS; (b) The registration output by RDA shows the robustness of RDA 12
5. Experimental Test Data and Registration Output 13
6. Adaptive Variance and Mixed Strategy 14
7. Registration by Composition 15
8. Examples of the Automatic Data Alignment 16
9. RDA Algorithmic Structure 16
10. Monte Carlo Simulation and Computational Speed 17
11. Outlier Rejection 18
12. Example of an Improved Registration by Data Refinement 19
13. Multi-layered Sensor Network: the highest level camera, denoted by L_1, monitors for large-structures with two UAVs, denoted by L_2 and L_3 22
14. (a) One-dimensional projection of the cost with respect to a scaling component around the ground truth. (b) Scaling factor versus the RDA performance results on 50 Monte Carlo simulations 24
15. Registration between a fixed reference and a frame in a video 26
16. Video Registration: Two UAVs generate video sequences, when they pass one of way points at two different times. In the bottom and left part, square marks represent feature points transformed into the base coordinate 27
17. A persistent sensor network includes layered sensing and multi-modal sensing 30
18. An Example of Data Registration between an Aerial Image from a UAV and a Google Earth Image. The rectified image frame of the aerial image appears on the top of the Google earth image by affine registration in (a) and by projective registration in (b). The projective map, as a result, better explains the relationship between the two images. 34
19. Four Dimensional Data Flow in UAV Sensing 35
20. Multi-scale Representation of an Image. Note that possible outliers like vehicles are averaged out at a lower resolution with a smaller image dimension. The number of features decreases in the order of 286, 100, and 29 35
21. Algorithmic Structure of ERDA 36
22. Registered Maps at Four Different Levels. The resolution increases, as the level decreases from the top left to the bottom right output 37
23. Registration at Two Different Levels 37
24. (a) The IR image is registered into the EO image. (b) Details of data refinement with multiple steps. The bottom right image represents a region above a threshold out of the transferred IR image into the base coordinate 38
25. Mosaic from 200 IR Image Frames for Persistent Tracking Applications 39
26. The Trajectory of Image Centers from 3750 EO Images 40
27. Rendering Pipeline 42
28. Estimated and Actual Locations 46
29. With More Refined Camera Parameters 46
30. Sample Image 1: Image frame 100 from Camera 0 49
31. Sample Image 2: Image frame 110 from Camera 0 49
32. Georeferenced Version of Sample Image 1 50
33. Georeference Version of Sample Image 2 50
34. Corner Features in Two Overlapping Images 53
35. Discovered Correspondence 53
36. RANSAC Hypotheses 54
37. 24 Correspondences 55

Figure **Page**

38. Two Rectified Aerial Images: Frame 68 and 568 from Camera 0 58
39. Examples of Correspondences 59
40. Frame 000003-003051 from CLIF 2006 Dataset 60
41. Warped 000003-003053 62
42. Georeferenced Frames with Errors 62
43. SIFT Alignment Translation 63
44. Total Alignment Error 64
45. CLIF 2007 Example Alignment Error 64
46. CLIF 2007 Absolute Alignment Error 65
47. Four Frames from a Stabilized Video 66

List of Tables

Table **Page**

1. List of Recovered Transformations 14
2. Computed Errors in Non-Georeferenced Images 56
3. Computed Errors in Georeferenced Images 57
4. Camera Parameters 60
5. Image to World Correspondences 61
6. Measured Error at Image Location (157, 89) 66

1. Executive Summary

This report covers the investigation by the Ohio State University CITR (Control and Intelligent Transportation Research) Laboratory team with PI: Professor Umit Ozguner, as part of the Revolutionary Automatic Target Recognition and Sensor Research (RASER) program.

The project, titled Robust Data Alignment (RDA) was initiated in April 2005 and continued through the summer of 2008.

In the third year of the project a task was added to provide an approach using state-of-the-art algorithms and software for Georegistration based georeferencing. This was oriented for specific data which was provided as a test case.

The technical approach of RDA supports Layered sensing and Multi-modal sensing in a Persistent Sensor Network where multiple unmanned aerial vehicles (UAVs) combined with stationary sensors are under operation for successful aerial tracking and surveillance missions. This final report presents an overview of our information-theoretic cost functional and its applications for a collaborative and distributed sensor network. Data registration with multiple UAV sensing can be useful for aerial monitoring and tracking systems within a dynamic sensor network.

2. Introduction

Multi-UAV sensing combined with stationary sensors has the potential of creating smart airborne systems for completing aerial tracking and surveillance missions over a determined area. However, understanding what is happening on the determined area is very challenging in general.

UAV imageries contain valuable information. The goal of RASER project is to develop a new methodology to extract a reference source for high-level tasks such as change detection from a registered reference and automatic target recognition. Precision registration is required done for sensing persistently. Hence, we start to formulate the problem of data alignment where we focus on invariant features of the UAV imageries, model their uncertain-feature spaces, and construct a new matching algorithm called robust data alignment (RDA). RDA relies on an information-theoretic cost criterion, whose optimal solution works for feature-based and correspondence-less data registration.

Sometimes, the presence of large motion in a spatial-temporal domain and/or dominant/dynamic outliers in feature space makes it very difficult to find an optimal transformation between two different images. Hence we demonstrate experimental results on many cases: video registration with a single UAV, weighted feature-based registration with multiple mini-UAVs, and registration with data refinement for multi-modal data registration. In addition, we address challenging issues with persistent sensing and formulate three problems: the problem of layered data registration, the problem of multi-modal data fusion, the problem of mosaic construction.

Georeferencing with reliable information could lead to a precise registration; however, due to the inaccuracy of intrinsic data including metadata of mobile sensors and geodesic data, we found some issues on registration with georeferencing on Columbus Large Image Format (CLIF) 2007 dataset. For registration without georeferencing, a two-image Scale-invariant Feature Transform (SIFT) example on CLIF2007 has been taken with the issues being encountered when using georeferencing and SIFT on CLIF 2006 dataset.

This final report is organized as follows: In Chapter 3, we describe the problem of data alignment. To solve the problem, we develop an information-theoretic cost criterion that models our feature-based and correspondence-less approach. This work also appears in [1], [2]. In Chapter 4, as in [3], we extend our previous work into multiple UAV sensing. Layered sensing with scaling factor has been investigated. Our previous cost criterion has been updated with more weights on stationary features in a video stream for registering two different video data. In Chapter 5, as in [4], we address problems in a persistent sensor network. Study on scale space tells that another representation of an original image can lead a more efficient solution for multi-level data registration. Our preliminary approach to multi-modal data registration contribute for all-time sensing when complementary information from different types of sensors need to be combined together. In Chapter 6, Georegistration has been applied for CLIF 2007 data. Inertial Measurement Unit (IMU) data of cameras attached on UAVs can lead to another version of aerial images: georeferenced images. However, due to an inherent error in metadata, Georegistration requires image to image registration. Registration with SIFT and Random Sample Consensus (RANSAC) algorithms also lead to a possible solution. This approach requires of recovering corresponding pairs of features between two different images. In the last

Chapter, we conclude our work in data registration. In Appendices, we describe our codes for robust data alignment.
Publications based on this project are:

1. S. Jwa and Ü. Özgüner, "Problems in Data Registration for Persistent Sensing," SPIE Defense & Security, March 16-20, Orlando, FL, 2008
2. S. Jwa, Ü. Özgüner, and Z. Tang, "Information-Theoretic Data registration for UAV-based Sensing," IEEE Trans. on Intelligent Transportation Systems, Vol. 9, No. 1, March 2008, pp. 5-15.
3. S. Jwa and Ü. Özgüner, "Multi-UAV Sensing Over Urban Areas Via Layered Data Fusion," IEEE Statistical Signal Processing Workshop, August 26-29, Madison, WI, 2007, pp. 576-580.
4. S. Jwa, Z. Tang, and Ü. Özgüner, "Robust Data Alignment Based on Information Theory and Its Applications in Road Following Situation," IEEE International Conference on Intelligent Transportation Systems, Toronto, Canada, September 17-20, 2006, pp. 1328-1333.

3. Information-theoretic Data registration

3.1 Introduction

Data fusion has been a popular approach to improve the performance of intelligent transportation systems for aiding various problems such as congestion control, accident handling, guiding emergency crews, collecting a real-time traffic patterns, and providing a reasonable route for traffic planners.

Within a distributed sensor network, multiple UAVs can capture complementary and redundant information on a road. To refine sensor readings and recover a common reference, data alignment is a crucial step. Under various assumptions, scenarios, and objectives, many plausible algorithms can be found (see [5], [6] and reference therein) in computer vision area. Typically, the alignment is based on establishing point correspondence between the feature sets extracted from two images. However, because of the considerable sensor noise and feature extraction errors in the sensor data captured from UAVs, the problem of finding a nearly-perfect feature correspondence has been often intractable. In addition, the different characteristics of multi-modal sensors, with the lack of similarity measure between feature points, make it even more difficult to find the feature correspondence.

In this work, we aim for a robust data alignment method without the necessity of knowing the explicit pair-wise correspondence between two feature sets. In order to achieve this goal, we construct an information-theoretic cost criterion using the maximum likelihood principle. Moreover, we design a dynamic cost model with an adaptive variance that accounts for explicitly an average error bound or implicitly the degree of uncertainties. We demonstrate that the size of an error bound decreases as pairs of points are eventually converges to each other. An intelligent robot, using a shrinking error-bound or reducing the degree of uncertainty, can recover the best pair-wise match of feature points, even if it seems not the exact match to a human.

Based on this new cost criterion, the problem of data alignment here becomes an optimization problem; that is, we seek an optimal parameter vector in the sense of minimizing the cost functional. To find a global solution of the information-theoretic cost functional, we consider a cooperative optimization scheme. In general, there is no single optimization scheme that guarantees convergence to a global minimum with a real-time operation; a global search scheme as a random search would reach a global minimum with a high computational effort, while fast search schemes such as gradient-based or direct search fail to reach a global minimum. This motivates cooperative optimization by mixing a global search scheme with a local, but fast, search scheme. There are many strategies with which local and global searches can cooperate. In [7], we see a case of a globalized Nelder-Mead method that achieves a required globalization by probabilistic restart. In this paper, we consider a series of searches consisting of the Nelder-Mead simplex method and a random search. To help the simplex method avoid a local minimum, we take a better initial point selected from the random searches on a convex region with extra computational costs.

With the assumption that the true transformation connecting two aerial images can be adequately approximated by a certain parametric transformation, we use an affine transformation of six

parameters for our experimental UAV data sets. However, without loss of generality, our RDA can be applied to any finite parametric transformation in a different scenario.

More interestingly, based on our Monte Carlo test in Section 3.4.3, we suggest a data refinement or outlier rejection method. Our comparison of performances, before and after data refinement, shows that we can obtain a more accurate affine map from refined data sets, when outliers such as moving vehicles on a road are detected and removed. With successful data fusion, we can use the registered map for many applications in intelligent transportation systems.

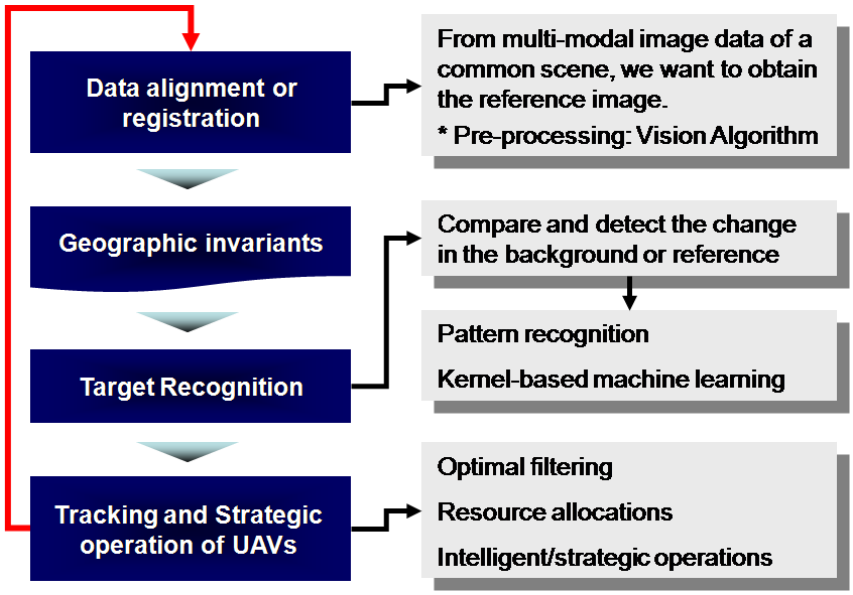


Figure 1. Integrated Process: target recognition, tracking, and monitoring tasks can utilize the geographic or reference map generated from data registration

For an integrated and iterative process as in Figure 1, the registered map will be useful for detecting moving targets that can be tracked by multiple UAVs under a certain strategic operation. Each block in this process is involved with rich research areas. Our ultimate goal is to use the registered ground for higher level tasks, such as tracking and monitoring, under a distributed multi-sensor network system.

The remainder of this chapter is organized as follows. In Section 3.2, we describe a general data alignment problem. In Section 3.3, we construct an adaptive cost criterion, develop a search strategy of finding optimal parameters in the data alignment problem, and evaluate the cost and the search strategy with a numerical stability test. The main theorem says that the cost criterion derived from the maximum likelihood principle is closely related to the entropy in information theory. More interestingly, the cost criterion is bounded from the above by the average of the squared Mahalanobis distance (see more details in [1]). In Section 3.4, we demonstrate the experimental results on data sets collected from a UAV experiment, supported by the U.S. Department of Transportation.

3.2 Problem Formulation

In this section, we describe the problem of data registration that corresponds to establishing a common reference, given a set of data, by finding a suitable transformation between two different images. Italicizing our approach, we can generally classify the problem into the following:

- correspondence-based versus *correspondence-less*,
- *parametric* versus nonparametric,
- intensity-based versus *feature-based*, and
- *mono-modal* versus *multi-modal* image registration

The common goal is to minimize a cost or to maximize a similarity measure.

3.2.1 Data Alignment Problem

Let I and $\tilde{I}$ represent a reference and sensed image, respectively. We assume that the two different images I and $\tilde{I}$ contain essentially the common scene as a basic requirement for image alignment. Let X and $\tilde{X}$ be two finite subsets, representing the feature points in I and $\tilde{I}$ with $|X| = N$ and $|\tilde{X}| = \tilde{N}$. Our goal is to recover an optimal function ζ between the two data sets.

Consider the following feature-based registration problem.

Problem: Given a cost function $\mathcal{J}$ and two feature data sets X and $\tilde{X}$, find a transformation $\zeta : \mathbb{R}^d \to \mathbb{R}^d$ that minimizes $\mathcal{J}[\zeta] := \mathcal{J}(X, \zeta \circ \tilde{X})$.

Following the information theoretic approach in [8] and using information theory in [9], we treat a set of feature data in an input or sensed image as the realization of random variables. Then we assume the following dynamic equation:

$$X_{k+1} = \zeta_k(X_k) + w_k, \tag{1}$$

where ζ_k and w_k are a transformation and disturbance, respectively, at a time k.
Our approach can use any parametric transformation for ζ in Equation (1), but in this paper we choose an affine transformation, i.e.,

$$\zeta(x) = Ax + B \equiv \begin{bmatrix} \theta(1) & \theta(2) \\ \theta(3) & \theta(4) \end{bmatrix} x + \begin{bmatrix} \theta(5) \\ \theta(6) \end{bmatrix}$$

$$\equiv [\ \theta(1)\ \ \theta(2)\ \ \theta(3)\ \ \theta(4)\ \ \theta(5)\ \ \theta(6)\ ]. \tag{2}$$

In this work, we assume the smooth motion of airborne mobile sensors like UAVs, flying high for the planar views of a scene, which generate a sequence of unknown affine-like transformations denoted by $\{\zeta_k; k = 1, 2, \ldots\}$ for a sequence of consecutive images denoted by $\{I_k; k = 0, 1, 2, \ldots\}$.

In recovering the transformation matrix, $A \in \mathbb{R}^{2\times 2}$, and the translational vector, $B \in \mathbb{R}^2$, in Equation (2), we do not assume any prior knowledge on feature correspondences. Instead, the probability distribution functions representing feature data sets are associated statistically. This association is the basic idea of the correspondence-less method. Our approach is comparable to the existing correspondence-less methodologies in the following subsection.

Another extension of the affine transformation is a projective transformation described by

$$\zeta(x) = H^T x \equiv \begin{bmatrix} \theta(1) & \theta(2) & \theta(3) \\ \theta(4) & \theta(5) & \theta(6) \\ \theta(7) & \theta(8) & 1 \end{bmatrix} x \tag{3}$$

This sort of extension helps when the affine map cannot be assumed, but the line is still preserved. Later in Section 5.3.1, we will show the case of registering Google Earth images and UAV images. Compared to the affine case, however, this extension can be highly sensitive with a choice of an 8-D initial parameter vector and computationally more expensive. As described below, this projective transformation is nonlinear.

$$x_{k+1} = \frac{\theta(1)x_k + \theta(2)y_k + \theta(3)}{\theta(7)x_k + \theta(8)y_k + 1},$$
$$y_{k+1} = \frac{\theta(4)x_k + \theta(5)y_k + \theta(6)}{\theta(7)x_k + \theta(8)y_k + 1}.$$

3.3 Cost and Search Strategy

In this section, we construct the following cost from the maximum likelihood principle:

$$J(\theta) = -\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \log \left\{ \frac{1}{C} \sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma^2} \|\hat{x}_i - x_j\|^2 \right) \right\}, \tag{4}$$

where, as in Section 3.2.1, $|X| = N,\ |\tilde{X}| = \tilde{N}$, and $\hat{x} = \zeta(\tilde{x})$ is a transformed feature point by ζ. Both $C = 2\pi\sigma N$ and σ are constant with this initial criterion; we have observed that the performance result is sensitive to the constant parameter σ, requiring different constants for different data sets. So, it was almost impossible to fix a universal constant value σ for various feature sets. One of the fundamental reasons for this difficulty could be explained in terms of uncertainties; in a dynamic environment, different feature data would behave as a random variable.

In order to overcome this issue and to obtain meaningful results, we seek an adaptive method to determine the value of σ_k at the k^{th} iteration. That is, we have

$$J_{\sigma_k}(\theta) = -\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \log \left\{ \frac{1}{C_k} \sum_{j=1}^{N} \exp\left(-\frac{1}{2\sigma_k^2} \|\hat{x}_i - x_j\|^2 \right) \right\}, \tag{5}$$

where $C_k = 2\pi\sigma_k N$.

3.3.1 Cost Criteria

First, we find that our cost model is indeed closely related to an important quantity in information theory. Let $\{x_i; i = 1, 2, \ldots, N\}$ and $\{\tilde{x}_i; i = 1, 2, \ldots, \tilde{N}\}$ be the sample sets of independent realizations drawn from random variables X and $\tilde{X}$, respectively.
Let $P_\theta(X) := P(\hat{X}),\ \hat{X} = \zeta(\tilde{X})$, denote a parametric version of $P(X)$. Then the maximum likelihood estimator wants to maximize $\prod_{i=1}^{\tilde{N}} p(\hat{x}_i)$. Taking the negative logarithm and normalization, we have the following:

$$J := -\frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \log p(\hat{x}_i) \approx \mathbb{E}_X\{-\log\ p(\hat{X})\} = -\int p(x) \log p(\hat{x})\, \mathrm{d}x \tag{6}$$

$$= -\int p(x) \log \frac{p(\hat{x})}{p(x)}\, \mathrm{d}x - \int p(x) \log p(x)\, \mathrm{d}x = D(X\|\hat{X}) + H(X). \tag{7}$$

Note that the cost J in Equation (6) represents the sample average of $\mathbb{E}_X\{-\log\ p(\hat{X})\}$. Clearly, we see the equivalence in Equation (7) between the MLE and the minimizer of the entropy, since $P = P_\theta$ a.e. implies that the relative entropy $D(X\|\hat{X})$ equals zero.
Next, using the Parzen-estimator in [10], we obtain

$$p(y) \approx \frac{1}{N} \sum_{j=1}^{N} G(y - x_j), \tag{8}$$

where $\{x_j : j = 1, 2, \ldots, N\}$ denotes a set of realizations of the random variable X, and G represents the so-called Parzen-window; here, we use a Gaussian kernel

$$G(x) := \frac{1}{2\pi|\Lambda|^{-1/2}} \exp\left(-\frac{1}{2} x^T \Lambda^{-1} x \right)$$

with a circular symmetric covariance matrix $\Lambda = \sigma^2 I$. Note that the Gaussian density function G in Equation (8) can be replaced by the alternative such as a Cauchy density function. Finally, as the main result of this section, we present the following theorem for our information-theoretic cost model.

Theorem:
Let X and Y be random variables. Let $\{x_i; i = 1, 2, \ldots, N_x\}$ and $\{y_i; i = 1, 2, \ldots, N_y\}$ be collections of independent feature points from X and Y, respectively. Then the maximum likelihood estimation for the parameter θ of an optimal transformation $\zeta(\theta)$ from Y into X is equivalent to minimizing the sum of the relative entropy and the entropy:

$$J(\zeta) := -\frac{1}{N_y}\sum_{i=1}^{N_y}\log\left\{\frac{1}{N_x}\sum_{j=1}^{N_x}G(\hat{x}_i - x_j)\right\} \approx D(X\|\hat{X}) + H(X),$$

where $\hat{x}_i = \zeta(y_i), \hat{X} = \zeta(Y)$, and G corresponds to the Parzen-window. Moreover, when $X = \hat{X}$ a.e., we have the relationship between J and the entropy: $J \approx H(X)$.

Regarding the duality between the relative entropy and the maximum likelihood principle, in 1973, Akaike [11] showed that maximizing the expected log likelihood ratio in maximum likelihood estimation is equivalent to maximizing the Kullback relative information. We believe that our cost criterion supports this duality principle.

Now, returning to our earlier plan on an adaptive method to deal with the sensitivity of the parameter σ in (4), we take the following adaptive rule:

Let x_i be any point in X. Then we define a degree of closeness, in terms of a distance measure, by

$$\sigma_i := \text{dist}(x_i, \hat{X}), \quad i = 1, 2, \ldots, N$$
$$\sigma_i = \min\left\{\|x_i - \hat{x}_j\| : \hat{x}_j \in \hat{X}, \quad j = 1, 2, \ldots, \tilde{N}\right\}.$$

The adaptive parameter σ_k in Equation (5), at the k^{th} iteration, is chosen by the average of the above distances; that is,

$$\sigma_k = \frac{1}{N}\sum_{i=1}^{N}\sigma_i. \tag{9}$$

Since we can consider this parameter as an error-bound on the average, we expect the size of this bound to decrease as we recover the transformation between two data sets. In Figure 2, we consider two cases: the first illustrates a good case where the error-bound shrinks and matching pairs, yellow circle as a transformed feature and red diamond as a true match, are recovered, while the second shows the other case where the error-bound does not shrink and the quality of alignment is not good. Dotted circles represent error-bounds at a time. Small circles and diamonds correspond to feature data in base and input image, respectively. On the base coordinate, the numbers represent transformed input feature data. Later, we will illustrate this phenomenon with some experimental results in Section 3.4.

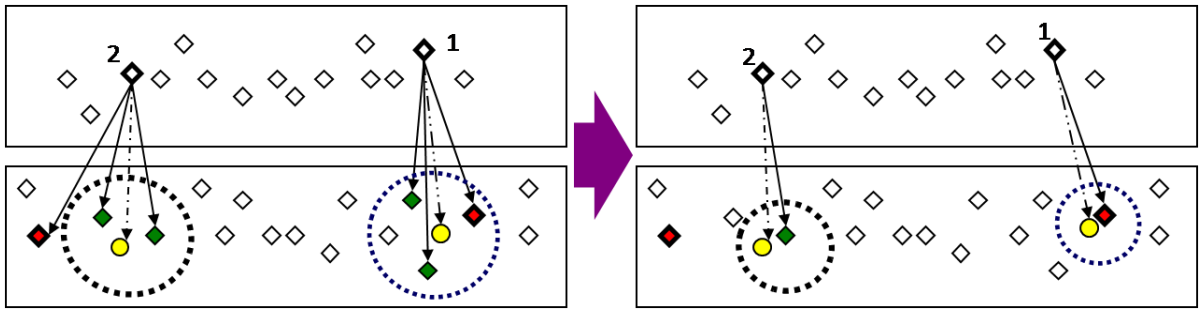


Figure 2. Two Cases of Varying Error-bounds in RDA

3.3.2 Search Strategies

Here, we focus on a mixed search strategy combining a stochastic search and the Nelder-Mead simplex method. By choice, we can start with either one in updating the six parameters in our linear transformation. The simplex method is one of the most popular direct search methods. This method may avoid a local minimum but has no theoretical support for convergence to a global minimum, as described in [12].

A stochastic search can be useful when we use a noisy cost instead of the true cost for recovering the true transformation. The following theorem, which appears in [13], supports the convergence of a stochastic search.

Theorem:
Suppose that θ^* is the unique minimizer of J in the sense that $J(\theta^*) = \inf_\theta J(\theta) > -\infty$, and

$$\inf_{\|\theta-\theta^*\|\geq\eta} J(\theta) > J(\theta^*), \quad \forall \eta > 0.$$

Let $S_\epsilon := \{\theta : J(\theta) < J(\theta^*) + \epsilon\}$ for any $\epsilon > 0$. If, for all k, there exists a scalar function $\delta(\epsilon) > 0$ such that $P(S_\epsilon) \geq \delta(\epsilon)$, then with noise-free cost measurements we have

$$\hat{\theta}_k \to \theta^* \ \text{ a.e.} \quad k \to \infty,$$

where $\hat{\theta}_k$ is an adjustment, at a time k, satisfying:

$$J(\hat{\theta}_k) < J(\hat{\theta}_{k-1}) < \cdots < J(\hat{\theta}_1) < J(\hat{\theta}_0) := J(\theta_0).$$

As discussed in Section 3.4, the purpose of a stochastic search is not to find a global minimum but to make the simplex method restart with a better initial for a global minimum, when the simplex method falls in any inappropriate local minima. Thus, the above theorem also supports our mixed search strategy.

The proposed mixed strategy belongs to a class of gradient-free methods, which is selected since we cannot guarantee whether the derivative of the cost J is unbiased. Moreover, this method makes it possible to use any non-differential cost functional that comes with a Cauchy density function instead of a Gaussian density function in the Parzen window method of the previous subsection.

Following is the summary of the cooperative or mixed search strategy. The initial and constant values below are used in our experiment in Section 3.4, but can be adjusted by choice.

Step 1: Choose an initial parameter vector. In our experiment, $\theta_0 = [1\ 0\ 0\ 1\ m_x\ m_y]$ with m_x and m_y representing the average translational movements in a horizontal and vertical direction, respectively. The initial standard deviation $\sigma_0 = 30$.

Step 2: Use the simplex method to find a minimizer until it satisfies a certain stopping condition: either the number of iteration should be less than 600 or the cost value is less than 10^{-5}.

Step 3: If $\sigma_k < 5$, then go to Step 4. Otherwise, by using the random search, find a better restarting parameter vector and go to Step 2. The number of searches is limited to 20,000.

Step 4: If $\sigma_k < 2$, then stop and return θ. Otherwise, go to Step 2.

3.4 Experimental Results

In this section, we consider a video sequence to demonstrate the results of our cost criterion and the mixed search strategy described in Section 3.3. Our collection of feature points is divided into the following two sets: manually selected sets having a small pixel error and automatically selected feature sets by the Kanade-Lucas-Tomasi (KLT) corner detector [14].

3.4.1 RDA vs. Correspondence-based Method

Consider two sets of feature points with no outliers and with a relatively small pixel error on the average. Note that we assume no knowledge of any feature correspondence initially between any pair of points. Experimenting on feature points selected by a human operator from a sequence of images as in Figure 3, we compare our correspondence-less method with one of typical correspondence-based methods, the least squares (LS) method. We find that the registered image by our approach is almost the same as the one by the LS approach, as shown in Figure 3.

In order to evaluate the performance of both methods, we compare the two approaches by testing the data sets extracted from the two images. Out of fifty pairs of feature points, forty pairs have been used as a training data set to find the invertible transformation and the others as a testing data set to evaluate performance measure such as an average error in pixels. This method of evaluating the alignment accuracy is known as the test point error method [6]; for alternatives to this method, a consistency check with multiple cues and mean square error in control points could be considered (see also [6]). Here, in Figure 3, we demonstrate how well the forty pairs of points match and provide the final registered output by our approach. The difference in average errors evaluated on test data sets is within a sub-pixel distance: 1.9039 and 1.9312 pixels by ours and the LS, respectively. This sub-pixel difference also indicates that the assumed linear transformation can be feasible for a video sequence from a UAV experiment. Parameter vectors lie in a six dimensional real space, $\mathbb{R}^6$, explaining the motions such as rotation, translation, scaling, and shearing. As described in Section 3.3.1, the result shows that we can recover not only the transformation but also the correspondence.

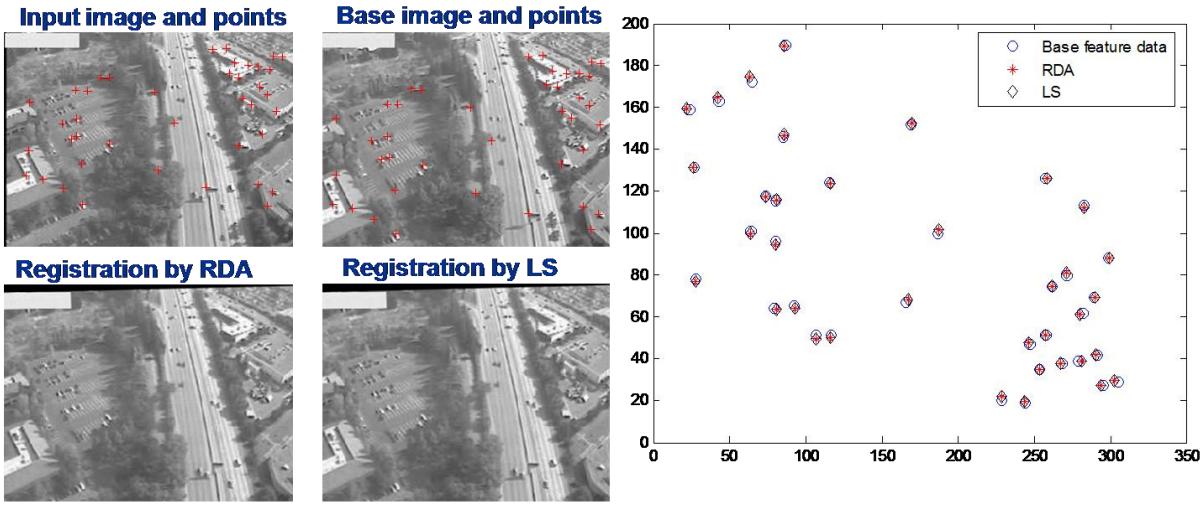


Figure 3. Two data sets are marked as red (+) marks and registered outputs are shown; without noise both RDA and LS give almost the identical results

Furthermore, with 5 noisy points (about 10% in each image) uniformly generated in each image, we find that our approach is more robust than the correspondence-based one, as shown in Figure 4 for the data sets in Figure 3. Performance evaluated by the same testing data sets shows that our approach outperforms the other: the average error by our approach is 1.9339 pixels, whereas the average error by the LS is 11.3368 pixels. The error value of this LS could be improved by a complex computation of the transformation as in [15], but generally feature data can be matchless (See Figure 8).

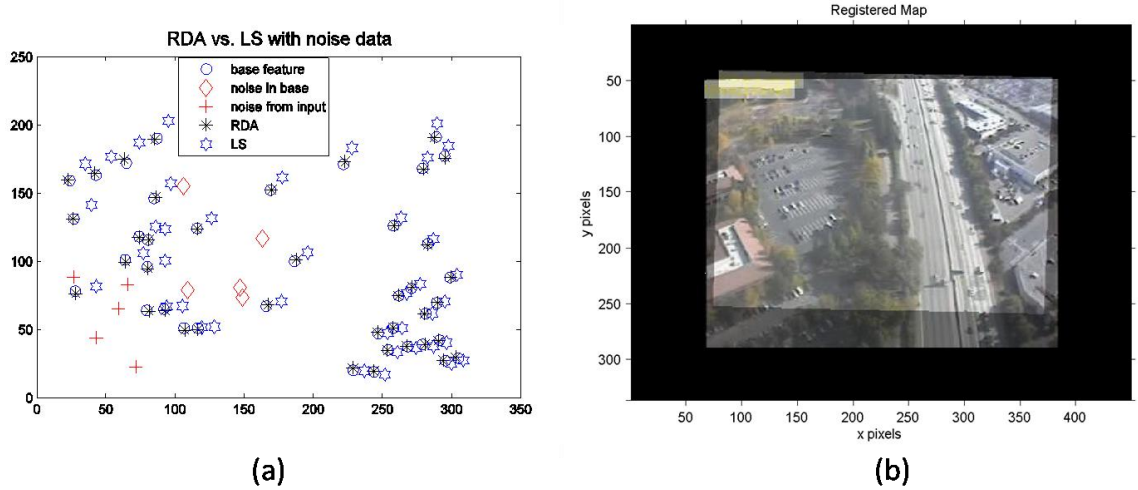


Figure 4. Robustness of the New Approach: (a) With noise we compare RDA and LS; (b) The registration output by RDA shows the robustness of RDA

3.4.2 Mosaic Image from a Video Sequence

In this experiment, we want to generate a new ground reference from a video sequence by a UAV. The mosaic image in Figure 5 is a new reference combining all the input images in the

test image set. The first frame is chosen as a base; others are transformed into the base coordinate by composition. RDA combines the test images into one mosaic map.

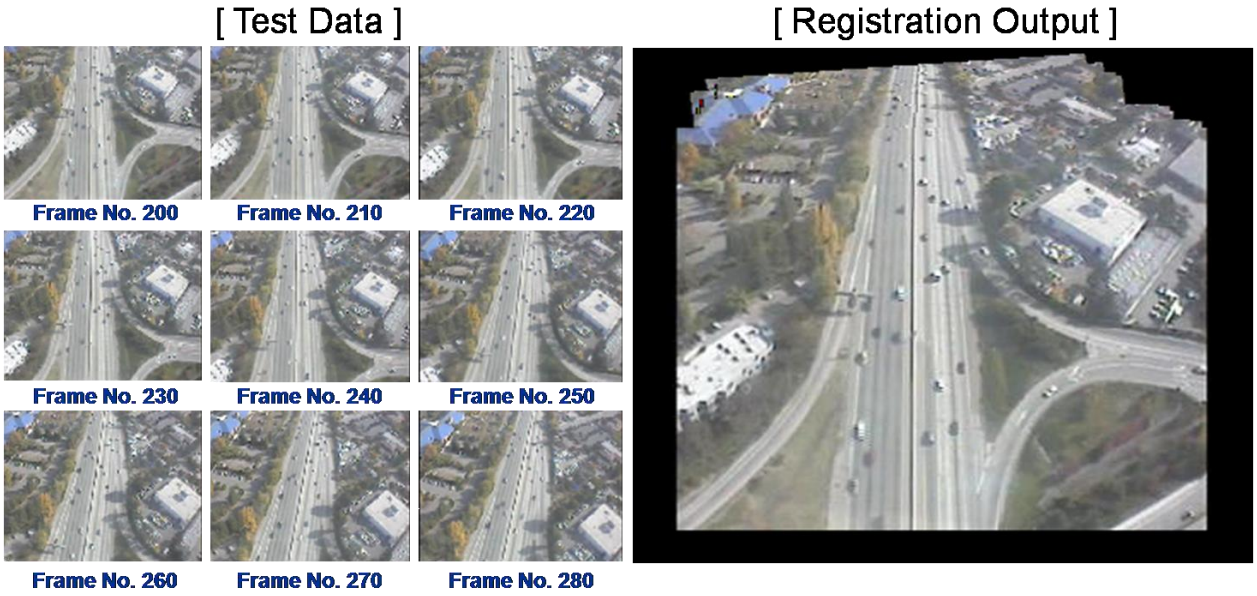


Figure 5. Experimental Test Data and Registration Output

The experimental result in Figure 6 sums up our approach: the right part illustrates that our mixed strategy effectively minimizes the cost, and, at the same time, the final value of the approximate variance within a sub-pixel confirms that the corresponding pairs are matched as described in Section 3.3, when the last image in the test set is registered to the eighth image in the test set as shown in the left part of the figure. The bottom part shows how the last image is registered to the first image. This large motion between the first and last image is recovered by the composition of eight transformations; that is,

$$\zeta^* = \zeta_1 \circ \zeta_2 \circ \zeta_3 \circ \zeta_4 \circ \zeta_5 \circ \zeta_6 \circ \zeta_7 \circ \zeta_8,$$

where $(\zeta_i \circ \zeta_j)(x) \equiv \zeta_i(\zeta_j(x))$. Here we have

$$\zeta^* = \; [\, 0.719 \;\; -0.062 \;\; 0.118 \;\; 0.508 \;\; 19.596 \;\; -26.639 \,]. \tag{10}$$

Note that we would have not achieved this result if one of the preceding eight transformations in Table 1 were incorrect. Figure shows how the last image is registered to the first image or the reference by ζ^* in Equation (10).

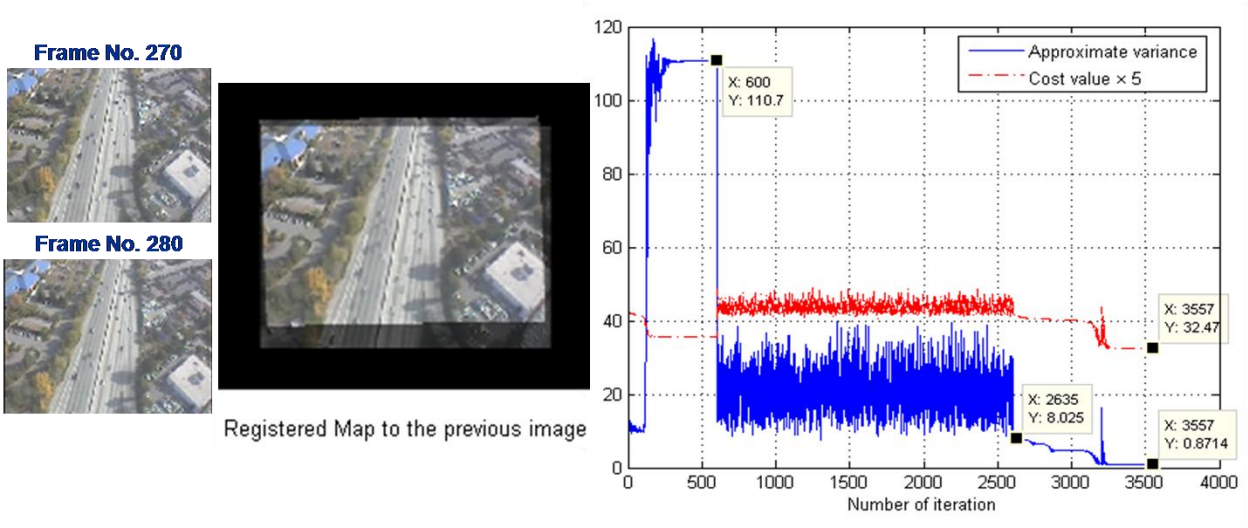


Figure 6. Adaptive Variance and Mixed Strategy

Table 1. List of Recovered Transformations

$\zeta_1 = [\ 0.9599 \quad 0.0152 \quad -0.0096 \quad 0.9185 \quad 9.8883 \quad -8.3944\ ]$
$\zeta_2 = [\ 0.9625 \quad 0.0099 \quad 0.0055 \quad 0.9155 \quad 4.4081 \quad -4.0453\ ]$
$\zeta_3 = [\ 0.9615 \quad 0.0010 \quad 0.0110 \quad 0.9212 \quad 1.8474 \quad -6.3338\ ]$
$\zeta_4 = [\ 0.9605 \quad 0.0226 \quad -0.0091 \quad 0.9202 \quad 10.0645 \quad -11.3158\ ]$
$\zeta_5 = [\ 0.9545 \quad -0.0953 \quad 0.1052 \quad 0.9168 \quad -19.3958 \quad 7.7142\ ]$
$\zeta_6 = [\ 0.9586 \quad -0.0106 \quad 0.0302 \quad 0.9195 \quad 9.4871 \quad -0.6945\ ]$
$\zeta_7 = [\ 0.9583 \quad -0.0122 \quad 0.0312 \quad 0.9198 \quad 4.4718 \quad -5.6837\ ]$
$\zeta_8 = [\ 0.9615 \quad -0.0150 \quad 0.0315 \quad 0.9248 \quad -0.4901 \quad -5.7265\ ]$

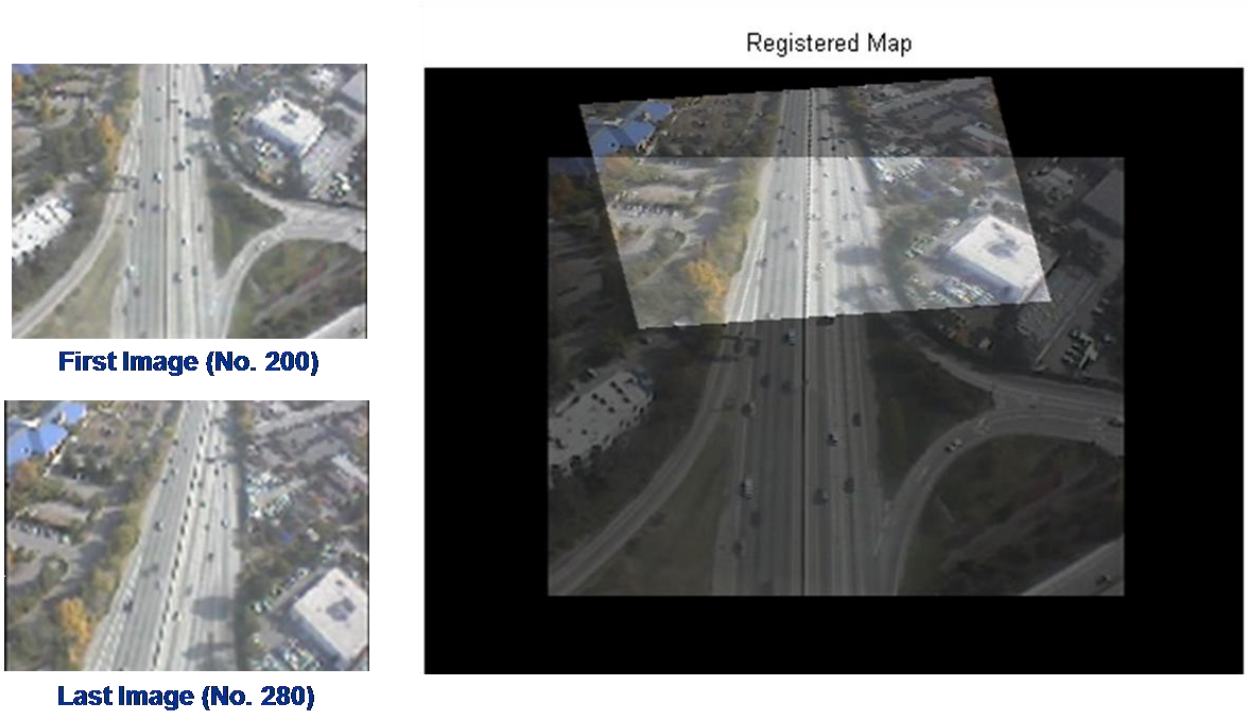


Figure 7. Registration by Composition

Next, we apply the proposed RDA method to the feature sets obtained from the KLT corner detector. This case is more challenging mainly due to the unexpected environmental condition and inappropriate feature data realized in the two data sets, even though the two images share a large region. As in Figure 8, with different numbers of feature points as $N = 91$ and $\tilde{N} = 148$, there exist outliers coming from the moving vehicles on the highway and matchless feature points from a non-overlapping region. Nonetheless the experimental result shows that the RDA method can successfully recover the transformation in spite of these types of noisy data. Note that the registered output clearly shows a good alignment of the highway, roads, and buildings when these types of feature data usually exclude a correspondence-based method for this quality alignment.

However, it is still needed to have a method of data refinement, as the second result in Figure 8 needs an improvement by rejecting some outlier-prone features. The more details on data refinement will be covered later in Section 3.4.4. The algorithmic structure of RDA is shown in Figure 9, where we have an additional step that eliminates outliers or refines the previous feature data.

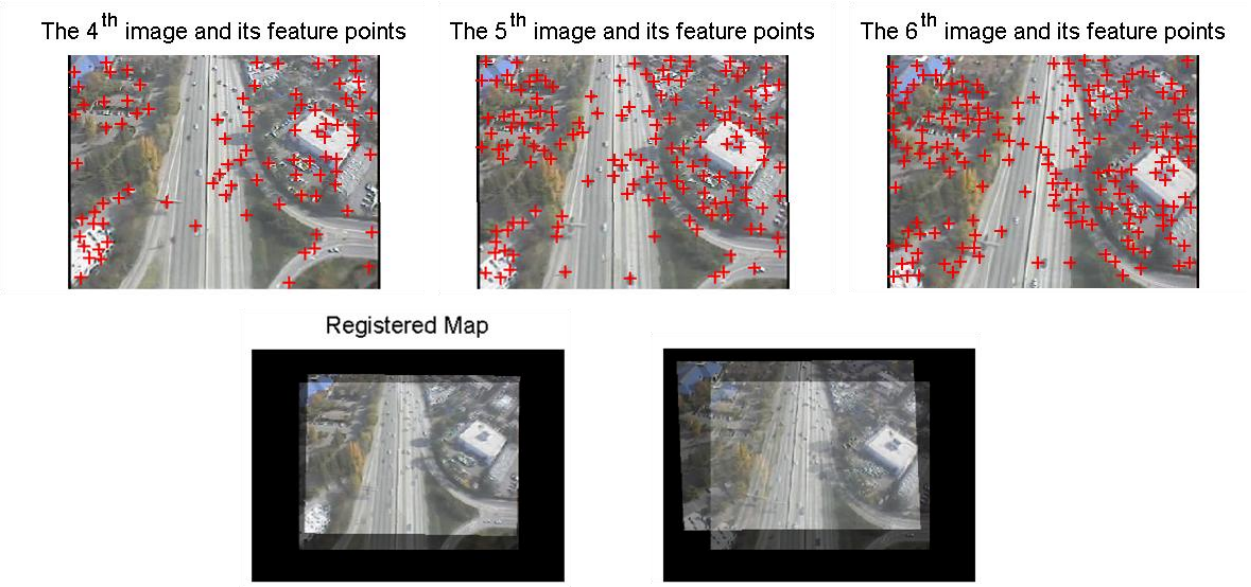


Figure 8. Examples of the Automatic Data Alignment

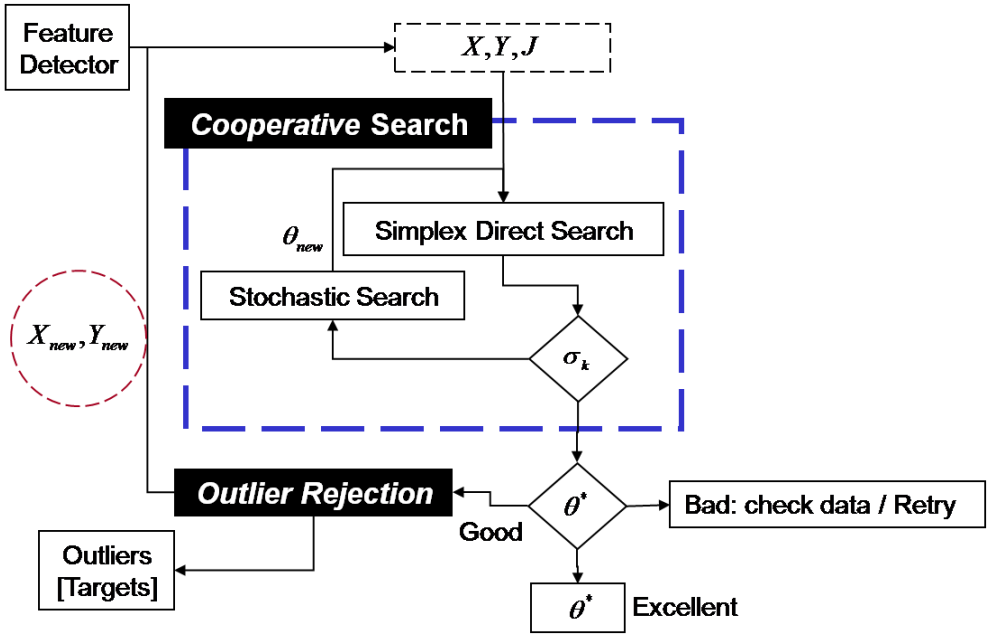
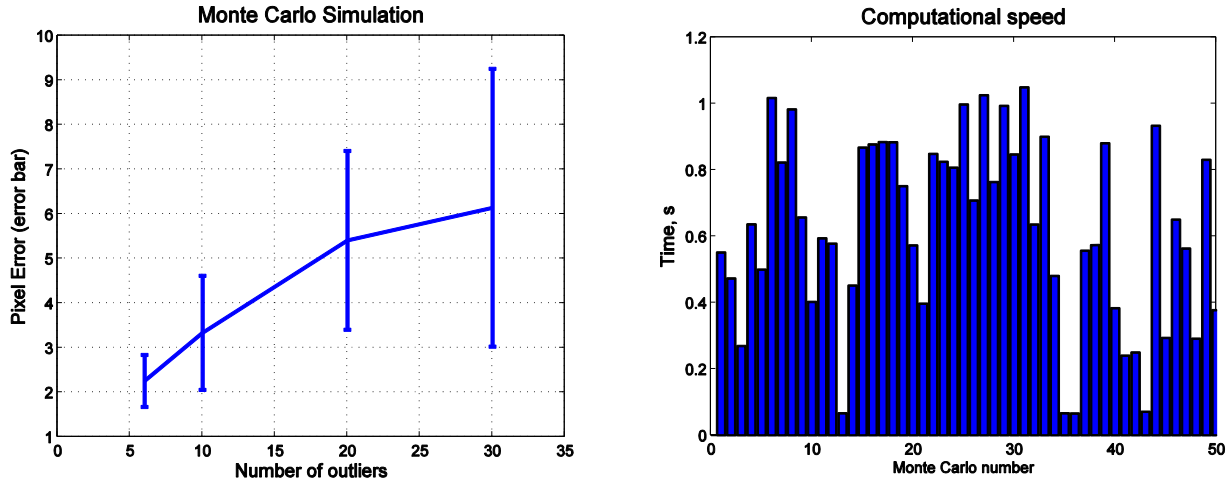


Figure 9. RDA Algorithmic Structure

3.4.3 Monte Carlo Simulation

To analyze how the RDA performs when a group of moving vehicles appears on a highway, we model them by generating noisy data from a uniform distribution on the highway. In Figure 10, we observe that with a higher rate of the outliers (on 40 inliers and 11 independent validating points) the error statistics get worse after the RDA algorithm. On the part (a), we consider different number of outliers and try 50 independent trials for each case. On the other part (b), we show a curve of computational speed in 50 trials on a Pentium IV 3 GHz computer: here 10

outliers are added. This simulation also explains why the automatic data alignment is more challenging as mentioned before.



(a) (b)

Figure 10. Monte Carlo Simulation and Computational Speed

3.4.4 Data Refinement

To improve the performance of the automatic data alignment, we suggest a method called data refinement (also called outlier rejection). In Figure 9, we have an additional step that can eliminate outliers or refine the previous feature data; our basic approach to detect a good feature is as follows: for a current iteration k and $i = 1, 2, \ldots, N$,
if $\sigma_k < \epsilon$ and $\|\hat{X} - x^i\| < \delta$, then x^i and the closest point in $\hat{X}$ indicate good feature points in the base and input, respectively. Here ϵ and δ are constants for small enough pixel error bounds. Remainders in the base and input correspond to outliers. For instance, Figure 4 illustrates that noisy points in both the base and the transformed feature data are matchless when our RDA finds the matching pairs for inliers.

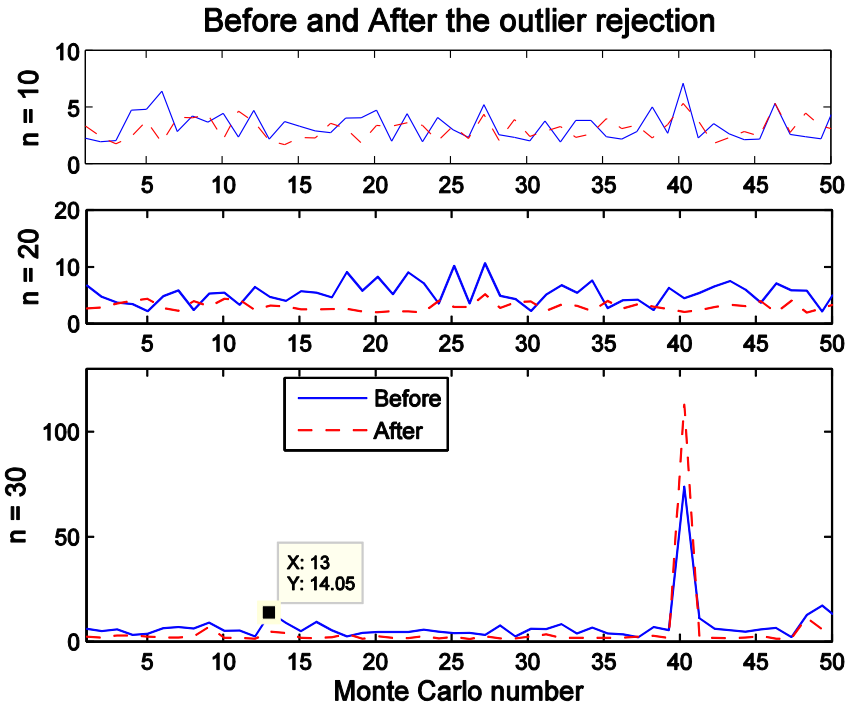


Figure 11. Outlier Rejection

In Figure 11, we compare average pixel errors before and after the outlier rejection method with 50 Monte Carlo simulations by increasing numbers of noisy feature data. We have a smaller pixel error after outlier rejection compared to the average error before outlier rejection. However, this outlier rejection method is feasible when a solution is close to a true one. Hence, we apply the outlier rejection method only if the error bound σ_k at an iteration index k is not too large. Otherwise, we see that we would accidently remove some good feature points and fail to get an optimal transformation. This phenomenon is shown in Figure 11 at the 40^{th} trial (n = 30). In this experiment, we call a transformation bad if σ_k is larger than 50.

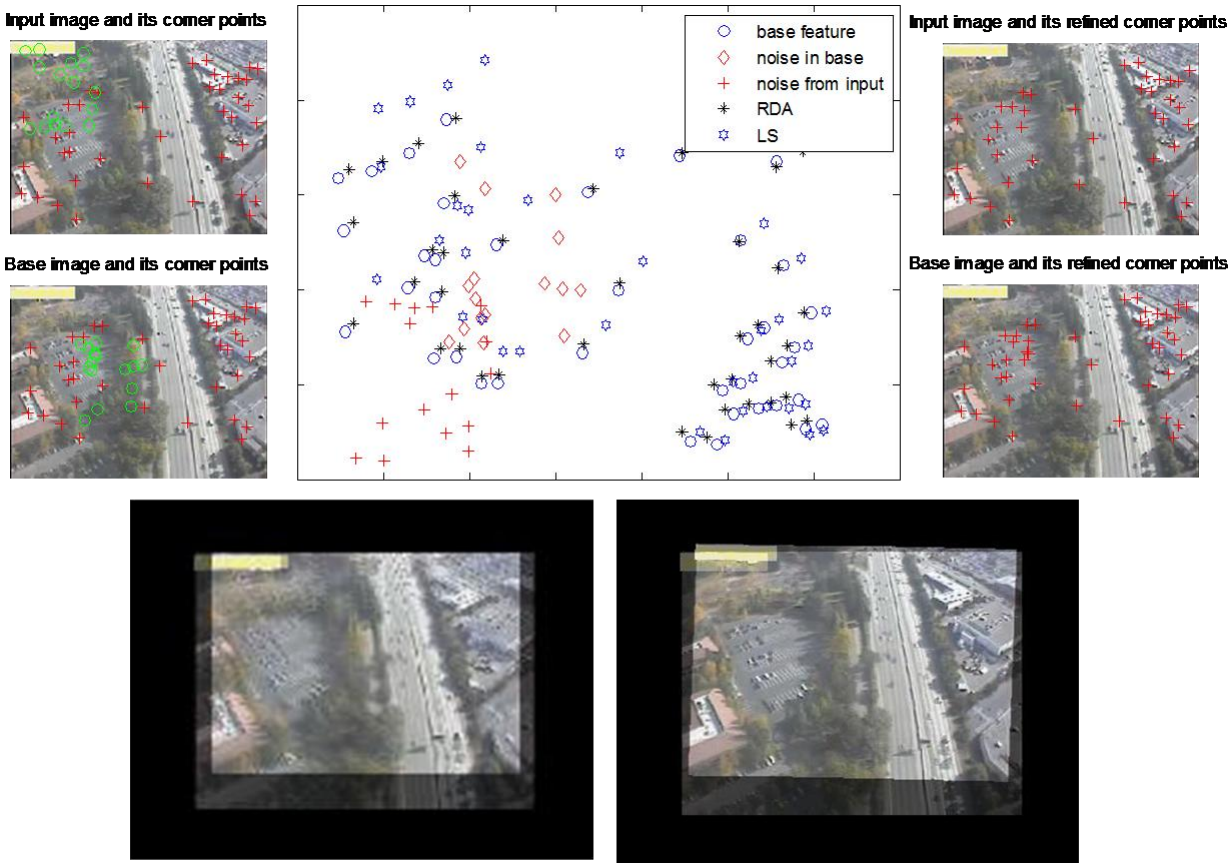


Figure 12. Example of an Improved Registration by Data Refinement

Figure 12 illustrates a good example of the benefit of outlier rejection: the error bound decreases from 7.386 to 1.602 so that the registered map with data refinement has a clearer alignment on the centerline of the highway and buildings. We can also consider another method to remove outliers. For instance, the RANSAC algorithm [16], which has been popular approach in computer vision, can be utilized; however, we do not consider it here to avoid an additional computational cost.

The outlier rejection block in Figure 9 makes it possible to localize target candidates in each image frame. In general, there is no guarantee of detecting all the targets; however, this coarse localization should be helpful for initializing target tracks for tracking.

4. Layered Data Fusion: Multi-UAV Sensing over Urban Areas

In this chapter, we address the problem of layered data fusion occurring within a collaborative and distributed sensor network. Multi-UAV sensing combined with stationary sensors has the potential of creating smart airborne systems for completing aerial tracking and surveillance missions over an urban area. However, matching two different data with multiple resolutions reveals the presence of large motion and dominant outliers, which makes this problem very difficult. Here we extend our earlier work to this type of multiple mini-UAV sensing by applying an information-theoretic cost criterion and cooperative optimization method. Assuming no zooming in the same type of cameras, we treat changes in scale-space as changes in UAVs' altitudes, and equivalently as changes in data resolution. Scale-space analyses indicate the importance of a prior adjustment of uncertain scaling factors. That is, to achieve a better performance in layered data fusion, it is critical to apply an approximate scaling factor first, within some error bounds, before applying a robust data alignment algorithm directly.

4.1 Introduction

Multi-level data fusion is a promising research area because many operational platforms consisting of mobile sensors from big aircrafts to small UAVs, including sensors attached on fixed platforms, collaborate for diverse applications such as surveillance [17], tracking, and traffic flow monitoring [18], [19]. However, recovering an optimal transformation between features of images from different cameras at different altitudes is nontrivial. This problem is referred to as multi-level/multi-resolution data fusion. Unlike the problems of a conventional image registration as in [6], there exist many different characteristics, since multi-level or layered images bring scaling factor, or equivalently resolution, issues.

First, multi-level or layered images create scaling factor, or equivalently resolution, issues. One image with a lower resolution can be obtained at a higher level/altitude and another image with a higher resolution can be obtained at a lower level/altitude. In this case, the fundamental assumption used in most of registration problems does not work. The overlapping region can be very small, which causes too many outliers. In order to find the right patch on the reference image by transforming the input image via an optimal map, we need to make a new assumption, such as having a close approximation of the initial scaling factor from any available ground sources or from the information on UAV dynamics, before comparing the area of a common region in both high and low resolution data.

Second, there exist few automatic feature extraction algorithms in the literature of image processing and computer vision for multi-level/multi-modal data fusion. We observe that in a high resolution data a normal corner finding algorithm as in [14] tracks vehicles whereas in a low resolution, or with a higher altitude for UAVs, it tracks the corner of stationary structures such as buildings. There have been some approaches to obtain a number of reliable corresponding points over a wide baseline [20] or with different resolutions [21]. However, it appears very challenging to obtain those reliable pairs from multi-modal data.

Third, a good model of transformation is not available initially. For ortho-rectified images, we might rely on an affine transformation, but when comparing a low-resolution image, say from a Google Earth map, and a high-resolution image from our video frames, it is required to have a good transformation model.

Hence, for a successful layered sensing, we have to overcome these nontrivial issues. Recently, in [2], the authors developed a RDA method as one of the key requirements for successful data fusion. They constructed an information-theoretic cost criterion and solved an optimization problem by a cooperative search strategy combining the Nelder-Mead simplex and stochastic search methods. This method belongs to the class of feature-based and correspondence-less approaches. In order to achieve automatic data alignment, unexpected outliers coming from moving entities such as vehicles and matchless features from any non-overlapping regions, must be treated in an efficient way, so that the UAVs can detect moving targets and provide a successful transformation simultaneously via the method of data refinement or outlier rejection [1]. This method seems beneficial not only for refining the previous feature data, but also for extracting moving entities (e.g. vehicles) contained in the outlier profile. To overcome the above issues in the problem of layered data fusion, we extend the RDA algorithm to include weighted feature points for video registration and multi-resolution/multi-level data registration.

In addition, the problem of registering a video sequence with a reference image is very challenging, as mentioned in [22], when any metadata such as sensor locations, telemetric information, and reference digital elevation models (DEMs) are not known. Our goal is to find the right patch on the reference image by transforming the input image via an optimal map in order to maximize the use of necessary information, contained in multi-level data, for the successful tracking of multiple moving targets.

The remainder of this paper is organized as follows. In Section 4.2, we introduce the problem of layered data alignment. Section 4.3 describes a mixed optimization procedure and scale space analysis. The sensitivity of our cost criterion with respect to a scaling factor is investigated with Monte Carlo simulations. In Section 4.4, we apply our layered data fusion technique to several examples: video registration with a reference image, and the alignment of two video sequences.

4.2 Data alignment in Multi-layered Sensor Networks

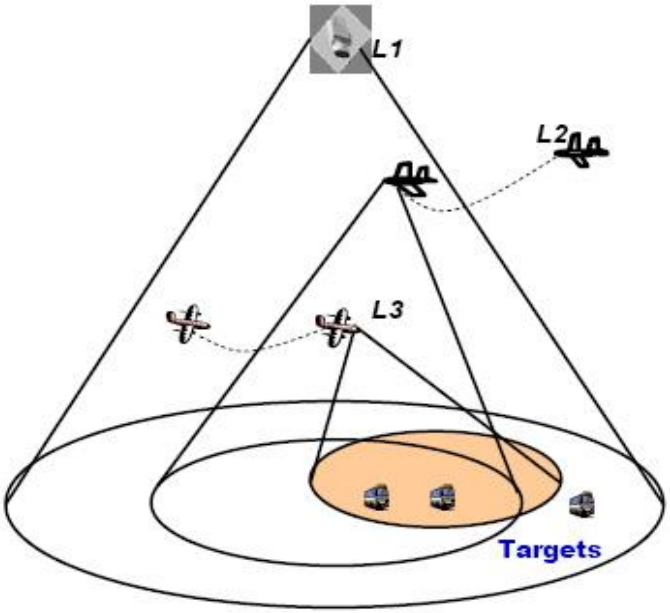


Figure 13. Multi-layered Sensor Network: the highest level camera, denoted by L_1, monitors for large-structures with two UAVs, denoted by L_2 and L_3

Let $\{I_i;\ i = 1, 2, \ldots, M\}$ be layered/sensed images from $\{L_i;\ i = 1, 2, \ldots, M\}$, where M is the number of different levels and L_i represents a space at the i^{th} level. Let F_i be a set of feature points in I_i with $|F_i| = N_i,\ \ i = 1, 2, \ldots, M$.

To find an optimal transformation between two different images, we first extract their feature data. Then a classical feature-based registration problem can be described as follows.

Problem:
Given a cost function $\mathcal{J}$ and two feature data sets F_i and F_j, find a transformation $\zeta : \mathbb{R}^d \to \mathbb{R}^d$ that minimizes $\mathcal{J}[\zeta] := \mathcal{J}(F_i, \zeta \circ F_j)$.

However, in our multi-layered network, the number of the feature data on non-overlapping region can be larger than one from overlapping region. So, it is required to examine a specific local region as a subset of the original region in a high level image.

Let $P \sim Q$ denote that two spaces P and Q have a sufficiently large common region that the above problem can be applied. Let L_{ij} have the following conditions:

$$1)\ i \leq j \quad 2)\ L_{ij} \subset L_i \quad 3)\ L_{ij} \sim L_j.$$

Finally, let F_{ij} be the feature data in F_i restricted on L_{ij}.

Now we define the problem of layered data alignment.

Problem:
Given a cost function $\mathcal{J}$ and two feature data sets F_i on L_i and F_j on L_j, find a transformation $\zeta : \mathbb{R}^d \to \mathbb{R}^d$ that minimizes $\mathcal{J}[\zeta] := \mathcal{J}(F_{ij}, \zeta \circ F_j)$.

Again, the transformation ζ can be any parametric function. Without loss of generality, we use an affine transformation with six unknown parameters first and use a projective transformation when the affinity does not work.
Our goal is to achieve a robust data fusion in this multi-layer setting.

4.3 Scaling Factor Analysis

For information-theoretical matching between two feature data sets, let us consider the following cost model in Section 3.3.1:

$$\mathcal{J} = -\frac{1}{N_y}\sum_{i=1}^{N_y}\log\left\{\frac{1}{C_k}\sum_{j=1}^{N_x}\exp\left(-\frac{1}{2\sigma_k^2}\|\zeta(\theta; y_i) - x_j\|^2\right)\right\}, \tag{11}$$

where σ_k is an adaptively determined variance at an iteration index k in a circular symmetric covariance matrix with $C_k = 2\pi\sigma_k N_x$; N_x and N_y are the numbers of feature points in the base and input data, respectively.

Let ζ^* be an optimal transformation between two feature data sets, F_i and F_j, in the following sense:

$$\zeta^* = \zeta(\theta^*), \quad \theta^* = \arg\min_{\theta} \mathcal{J}(\theta; F_{ij}, F_j), \tag{12}$$

where $F_j \sim F_{ij} \subset F_i$ is given as in Section 4.2.

To obtain an optimal parameter, we use the cooperative or mixed search strategy as in Section 3.3.2.

4.3.1 Scaling Factor Analysis

In this subsection, we investigate scaling factors when other motions are zero.

Let s be the scaling factor. Then the true transformation is $\theta^* = \begin{bmatrix} s & 0 & 0 & s & 0 & 0 \end{bmatrix}$.

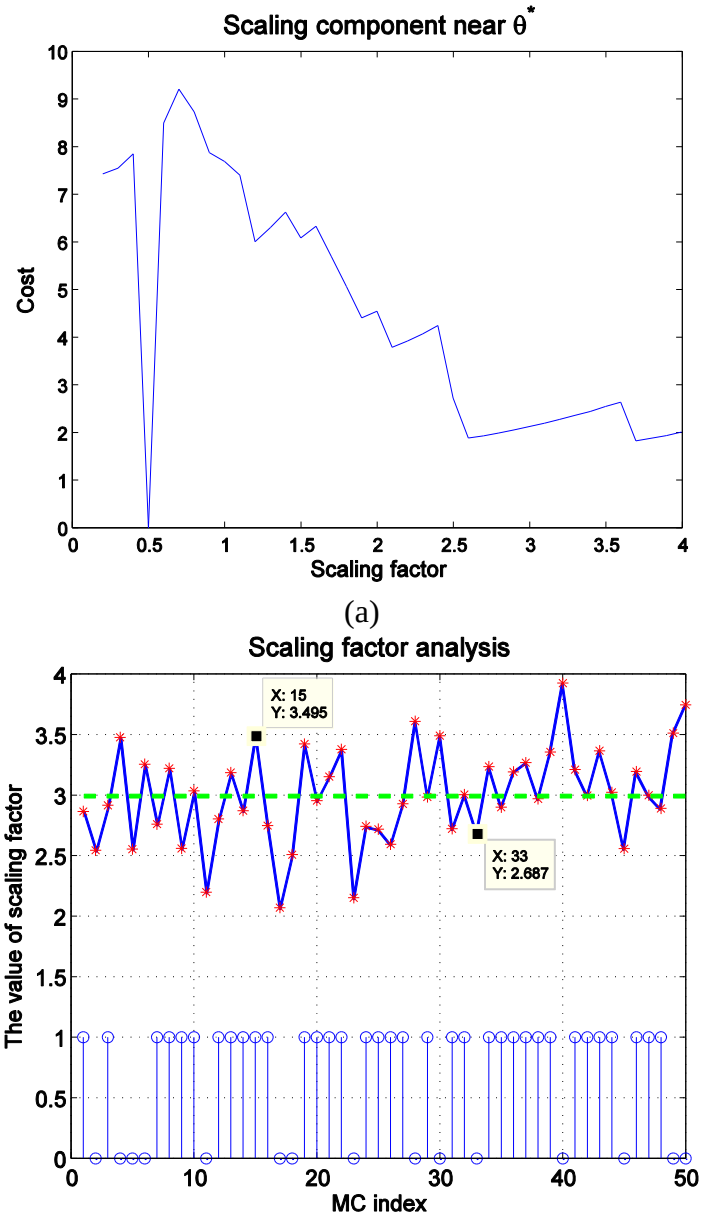


Figure 14. (a) One-dimensional projection of the cost with respect to a scaling component around the ground truth. (b) Scaling factor versus the RDA performance results on 50 Monte Carlo simulations

Figure 14 (a) shows the sensitivity of our cost criterion with respect to scaling factors near θ^*. In Figure 14 (b), we demonstrate the importance of choosing a good estimation to an unknown scaling factor. Here we randomly generate 50 points as the base features with 10 additional noisy data. Input data are obtained from a certain ground truth; then 50 Monte Carlo tests are performed to recover the ground truth. The top of

Figure 14 (b) shows the estimated scaling factors when the true scaling factor is 3. At the bottom, the outputs with the value of one imply that average pixel error was less than 5, which means successful cases. The output of value zeros represents failing cases. In this Monte Carlo simulation, we see that within an error bound, either ± 0.32 or 10%, of the true scaling factor, the overall alignment process works well.

4.3.2 Weighted Feature Points

Based on our observation of corner points obtained from general video sequences using the algorithm in [14], we can modify the original cost criterion (11) with weighted feature points. It is because corner points with higher eigenvalues, such as points from buildings, are more likely repeated in the following frames, unlike most of outliers. Hence, we can put more weights on those points assigned to stationary structures without any structure detection algorithm such as a building detection algorithm.

Now we have

$$\mathcal{J} = -\frac{1}{W_y}\sum_{i=1}^{N_y}\omega_y(i)\log\left\{\frac{1}{W_x}\sum_{j=1}^{N_x}\omega_x(j)G(\hat{x}_i - x_j)\right\}, \tag{13}$$

where $\omega_x(i)$ and $\omega_y(i)$ represent weights of x_i and y_i, $W_x = \sum_{i=1}^{N_x}\omega_x(i)$, $W_y = \sum_{i=1}^{N_y}\omega_y(i)$, G Gaussian kernel, the other terms being the same as before.

Note that this cost model does not affect the computational efforts of the original one.

4.4 Experimental Results

Considering a distributed sensor network consisting of multiple UAVs and a fixed sensor, in this section, we provide the following experimental results: registration on a single video sequence, stitching a single video frame on a given reference image, and the registration of two different video sequences.

4.4.1 Video Registration

Suppose that we have one reference with a low resolution and a sequence of input images with mostly higher resolution which can be registered in the reference. Our objective here is to generate a series of patches. This problem belongs to a class of one fixed and another mobile platform scenario.

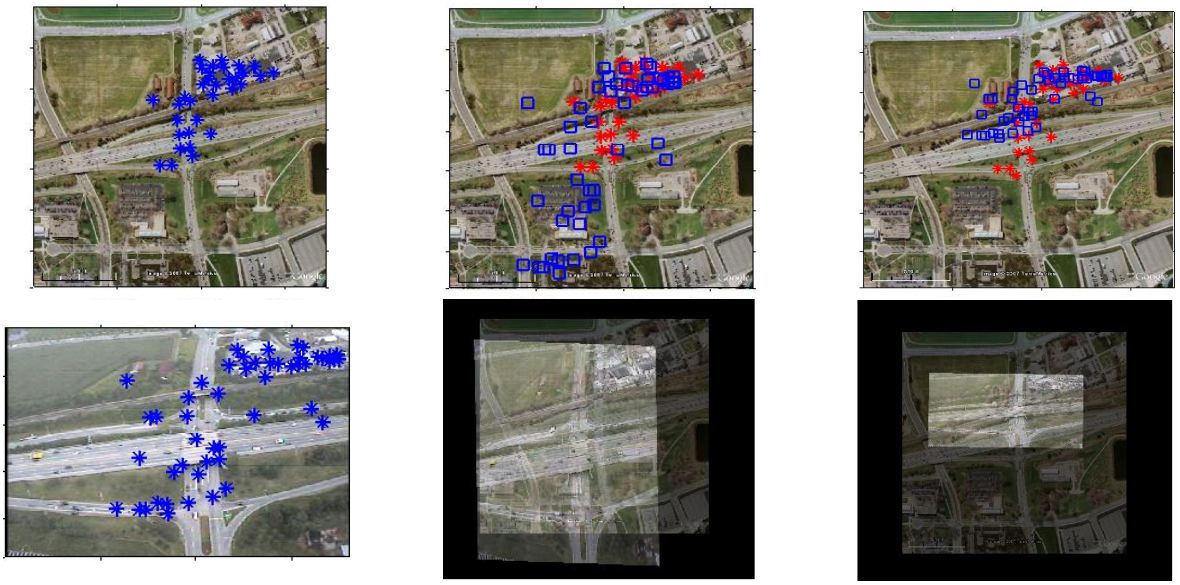
In Figure 15, the feature points on a reference image are chosen based on measurements from a video sequence. Note that in this image with a low resolution, vehicles on a highway rarely appear as corners, unlike the input image on the right side, which has a higher resolution. Figure 15 (a) shows that scaling factor is not recovered at all when we directly work on the feature data sets; however, Figure 15 (b) shows an improved result when we use an approximate scaling factor 2.37 and use weighted feature data. While an ortho-rectified reference image and an image from a UAV could not be explained by a single affine transformation, the transformed feature data, which is marked by squares, shows a reasonable group-correspondence with the feature data on the reference coordinate.

4.4.2 Simultaneous Registration and Vehicle Detection

In this subsection, we present a preliminary result by extending the previous internal registration of a single UAV case into multiple UAV cases. Here we have two UAVs monitor a highway and west campus area of the Ohio State University.

Both UAVs being assigned way points, one of which is identically put on the highway, we observe that some feature points from stationary structures, such as buildings and highways, are repeated in consecutive frames whereas moving feature points turned out not repeated as much. i.e., the higher the weights, the more likely the inliers are. So, in this part, we consider weighted feature data which is helpful to find an optimal transformation by reducing the current outliers-to-inliers ratio of more than forty percent.

Figure 16 illustrates that the same way point can make video registration possible with weighted feature data. The top of the figure shows the base and input images with their feature data and the bottom the matched feature data in the base coordinate with the registration output. With a more accurate video registration, both UAVs can cooperatively complete their missions. Some of moving vehicles can be identified as either unmatched '*' or unmatched '□' marks. The registration result in the bottom and right part on the video sequences indicates the potential of aerial monitoring and tracking systems.



(a) With a poor scaling factor (b) With a better scaling factor

Figure 15. Registration between a Fixed Reference and a Frame in a Video

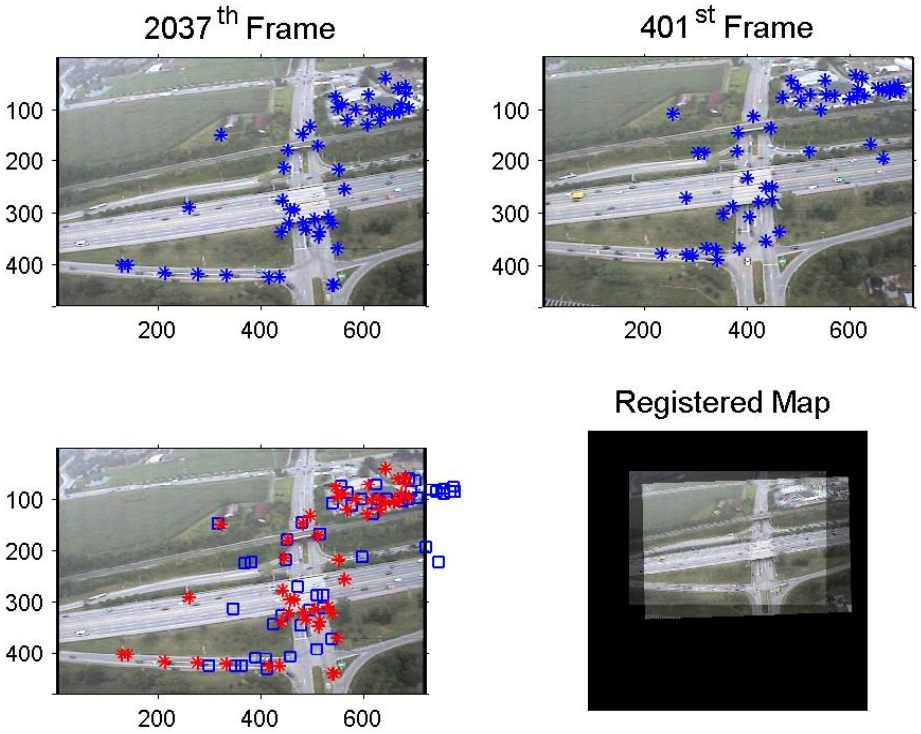


Figure 16. Video Registration: Two UAVs generate video sequences, when they pass one of way points at two different times. In the bottom and left part, square marks represent feature points transformed into the base coordinate

5. Problems for Persistent Sensing

Persistent sensing by Unmanned Airborne Vehicles (UAVs) has brought up challenging issues including multi-scale analysis, multi-resolution analysis, and scene localization. The multi-scale and multi-resolution issues occur when a mobile sensor changes altitudes or two different sensors with the same camera provide any redundant images from different altitudes. To overcome these issues, we first focus on collecting invariant feature data from the multi-resolution representation of a high resolution image. The relationship between multi-resolution and multi-level/altitude representations is investigated by the feature data. Recently, an information-theoretic matching criterion has been developed for robust data registration without any knowledge of feature correspondence. This criterion is used as an intelligent computing algorithm of choosing a good scale-representation that helps to find an unknown scaling factor between two different and redundant measurements. The last issue of scene localization is required for identifying the scene visited before. In this work, projecting the center locations of image measurements onto the two dimensional reference coordinate, we demonstrate the trajectory of the mobile sensor based only on the extracted transformations, not relying on any telemetric data of the mobile sensor which is not available persistently.

5.1 Introduction

The needs for persistent and ubiquitous surveillance are growing in many research areas such as sensor management in a battle field, traffic analysis [18] and management [19] over a road, and activity monitoring by airborne video registration [17], [22] over an urban area. With the advent of many different types of sensors, an enormous amount of imagery data will be available. They contain low (near ground) level and high-resolution imageries, high level and low-resolution imageries, and different types of information from different types of sensors.

In this work, we define persistent sensing by a mechanism that preserves the previous reference about a region of interest and update the current reference with a new measurement from any part of the region. This mechanism can be thought of as temporal ubiquity, while ubiquitous sensing can be as spatial persistence. Intelligent sensing requires persistent sensing with a structure that always preserves the previous version of itself after it changes. Hence, persistent sensing must provide a temporal data flow which is immutable. With no measurements, a simple persistence could be achieved by one reference; however, it is impossible and inefficient to rely on only one reference. A better method is to exploit the similarity between new and old information, assuming most measurements cause small changes to a current reference. For instance, given a series of image frames from unmanned ariel vehicles (UAVs), a temporal data flow can contain a finite number of versions that describe an aerial view of a scene, which can be constructed by a video mosaic [23] with a large view. The common region between multiple frames will not be duplicated, but will be shared between both the old version and the new version.

With this new concept, persistent sensor networks must complete the following tasks:

- the task of layered sensing
- the task of combining multi-modal sensor information
- the task of updating a previous reference with a new information

Recently, information-theoretic matching method [2] [1] has been developed. This method belongs to feature-based and correspondence-less approaches. Most feature based approaches in image registration [6] rely on the knowledge of correspondence, which is very challenging for feature data collected from fixed platforms and mobile platforms as UAVs and is almost impossible to obtain with different types of sensors. In this chapter, we further investigate the previous Robust Data Alignment (RDA) algorithm to multi-modal data alignment and also extend affine registration to projective registration. Affine registration is satisfactory with a smooth motion of a UAV, providing a plane-like scene. However, a projective transformation is needed in case of registering Google Earth images and aerial images, as you will see later.

Our ultimate goal is to provide an efficient algorithm, Extended RDA, to deal with the three tasks described above for persistent sensing. This Chapter is organized as follows. In Section 5.2, we introduce the three problem mentioned above for persistent sensing. Section 5.3 describes an extended RDA algorithm as a methodology to attack the three problems. The previous affine registration can be extended to projective registration, and multi-resolution approach can provide an efficient number of invariant feature data at a different scaling factor. In Section 5.4, we provide some experimental results on data sets supported by the DARPA video verification of identity (VIVID) project.

5.2 Three problems for Persistent Sensing

In this paper we want to attack the following three problems:

- Multi-level and multi-resolution data registration
- Multi-modal data registration
- Mosaic construction for scene localization

The first problem is related to so-called layered sensing, because different levels due to UAV altitudes' variation affect image resolutions/dimensions. We also need to consider multi-level and multi-resolution data structure analysis in spatial-time space. The major part is to extract invariant feature data at an optimal layer for their successful registration. So, layered sensing is one of the key contributors to persistent sensing.

The second problem occurs when different types of sensors are distributed for persistent sensing. For instance, we will consider data alignment between a pair of electro-optic and infrared images in Section 5.4.2. The major problem is to match the related information, even with asymmetric but complementary information. This multimodal data registration can help to achieve all-time-sensing.

The third problem of creating a mosaic needs to be solved for a highly efficient representation of a complex environment. We will see that the major question is how to minimize an accumulated error after the composition of a sequence of transformations from the current RDA algorithm. In Figure 17, we illustrate the three problems under a persistent sensor network. The network combines a layered structure with different modalities. A successful data fusion will lead to a high-dimensional reference with a layered information structure.

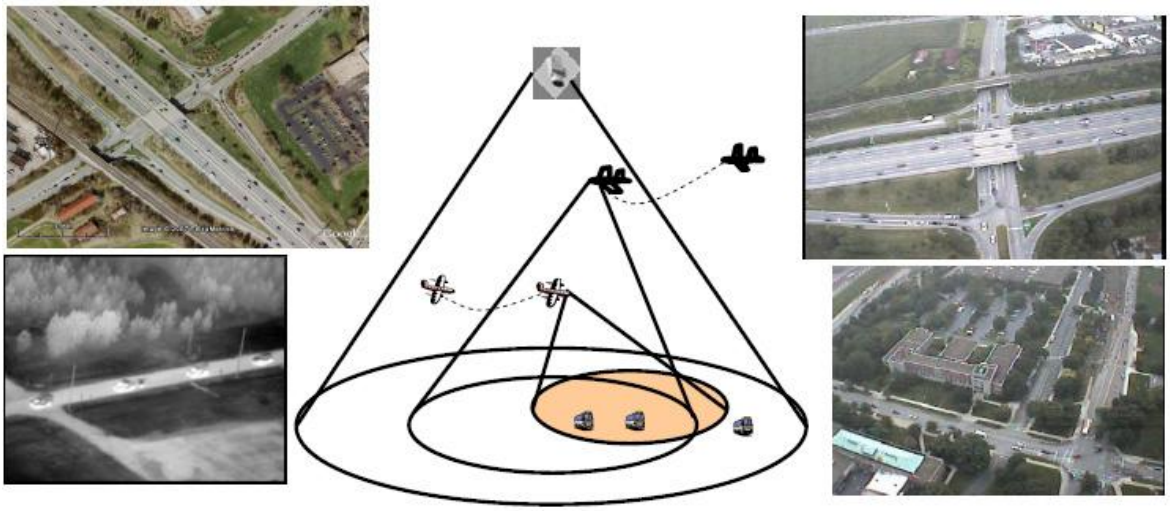


Figure 17. Persistent Sensor Network for Layered Sensing and Multi-modal Sensing

In the following subsection, we review scale space theory. Scale factors are naturally embedded in a transformation, as a crucial factor, when we recover the motions among multiple image frames. It is expected that our flying sensors keep changing its altitude within a certain range, which leads to a multi-level representation of a common scene. Hence, optimal processing of data alignment requires its multi-scale representation.

5.2.1 Scale Space Theory: Review

Let $I(\mathbf{x}) \in \mathbb{R}$ be an image, where $\mathbf{x} = (x\ y)' \in \mathbb{R}^2$. In scale space theory, scale space is a concept with a continuous parameter s. This concept later can be applied to generate a multi-scale representation of an image. Now, we study the definition of scale space and a scale space generating operator, and introduce linear and quadratic scale space.

Definition of scale space: a data structure that consists of a sequence of images with different resolution is known as a scale space; we write $I(\mathbf{x}, s)$ to indicate the scale space of the image $I(\mathbf{x})$.

In general, a scale space generating operator $\mathcal{B}$ should satisfy the following two requirements:

- Information-decreasing property: no new details must be added with increasing scale parameter.
- Semi-group or scale invariance property: the convolution kernel should satisfy

$$\mathcal{B}(s_1)\mathcal{B}(s_2) = \mathcal{B}(s_1 + s_2)$$

$$\mathcal{B}(s_2) = \mathcal{B}(s_2 - s_1)\mathcal{B}(s_1), \quad s_2 > s_1$$

It turns out that the Gaussian kernel is the only convolution kernel that meets the above requirements and is in addition isotropic and homogeneous.

Consider the following linear diffusion process:

$$\frac{\partial I}{\partial s} = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} = \triangle I,$$

$$I(\mathbf{x}, 0) =: I(\mathbf{x}).$$

The solution of the above process is known as linear scale-space [24] or Gaussian scale-space:

$$I(\mathbf{x}, s) = \frac{1}{4\pi s} \exp\left(-\|\mathbf{x}\|^2/(4s)\right) * I(\mathbf{x}), \tag{14}$$

where a standard deviation σ of Gaussian equals $\sqrt{2s}$. Therefore, by using Equation (14), we can generate a continuous family of images, $\{I(\mathbf{x}, s); s \geq 0\}$, from $I(\mathbf{x})$.

Assuming that we can treat the scale parameter s as the time, the following diffusion process can establish another scale space:

$$\frac{\partial I}{\partial t} = D_0 t \triangle I. \tag{15}$$

To show that the general solution of the diffusion equation (15) is equivalent to a convolution with a smoothing mask, we first apply the spatial Fourier transform. Then we have

$$\frac{\partial \hat{I}(k)}{\partial t} = -4\pi^2 D_0 t |k|^2 \hat{I}(k). \tag{16}$$

By solving the above equation, we obtain a general solution

$$\hat{I}(k, t) = \exp\left(-2\pi^2 D_0 t^2 |k|^2\right) \hat{I}(k, 0), \tag{17}$$

which is equivalent to

$$I(\mathbf{x}, t) = \frac{1}{2\pi D_0 t^2} \exp\left(-\frac{\|\mathbf{x}\|^2}{2D_0 t^2}\right) * I(\mathbf{x}, 0).$$

Putting $s_q = D_0 t^2$, it is called a quadratic scale space. We can also show the converse; that is, starting from the general solution (17) leads to the differential scale space. Another study of a class of nonlinear scale-spaces appears in literature [25] with the connection to generalized entropies.

Let us next study the following multi-grid representations [26]: Gaussian and Laplacian Pyramids. Firstly, each level of the Gaussian pyramid is obtained from the previous level by a low pass filter. The size reduction is done with a smoothing operation to generate its scale space representation as follows:

$$I^{(0)} := I(\mathbf{x}), \quad I^{(l+1)} = \mathcal{B}_{\downarrow 2} I^{(l)}, \quad l = 0, 1, 2, \ldots,$$

where $I^{(0)}$ is the zero level of the pyramid that is identical to the original image. For the $(l+1)^{th}$ level from the l^{th} level, the resolution decreases by a factor of two, and the size of the image decreases. The Gaussian pyramid is useful when we want to reduce the size of an image.

Secondly, the Laplacian pyramid represents a sequence of bandpass-filtered images. This efficient scheme for a bandpass decomposition of an image is achieved by

$$L^{(l)} = I^{(l)} - {}_{\uparrow 2}I^{(l+1)}.$$

Note that the Gaussian and Laplacian pyramids are effective but inflexible, since the scale parameter changes by a fixed factor of two; however, in some cases, a continuous scale parameter is preferred to produce a finer scale representation.

5.3 Methodology: Extended RDA

In our previous work [1] where we developed a new scheme called RDA, information-theoretic matching has taken the following steps:

Step 1: Extract an invariant set of feature data from each image and regard them as densities of the images
Step 2: Use the RDA algorithm to find an affine map between the images
Step 3: Separate features into the two classes: Good feature and outlier-prone feature
Step 4: Go to Step 2 and use the refined data from Step 3, unless a stopping criterion is satisfied.

Our progress towards Extended RDA (ERDA) consists of the following two parts: projective registration and multi-scale based feature selection. Firstly, in our previous work, we claimed that the RDA algorithm can handle a further extension into any parametric transformations. As an example, we are extending a previous affine map into another, a projective map. This sort of extension helps when the affine map cannot be assumed like Google Earth images and UAV images. Compared to the affine case, however, this extension can be highly sensitive with a choice of an 8-D initial parameter vector and computationally more expensive. Secondly, we are inserting a multi-scale or multi-resolution step to register any redundant images captured at different altitudes/scales. More importantly, when we deal with image data with large sizes and high resolutions, we need to convert them to ones with smaller sizes and lower resolutions to extract a sufficient class of features for feature-based registrations. The fewer features, the faster and more efficient the registration process. Hence, we are investigating a methodology to find such an efficient feature set over a scale space.

More details of the two parts are provided in the following subsections.

5.3.1 Projective Registration

The previous RDA algorithm has assumed an affine-like motion between two related images. An affine map transforms a parallelogram into another parallelogram. This assumption might be accepted practically; however, when the image grabber moves within an uncontrolled environment such as a strong wind, we see that the two images cannot be explained with the affine motion.

A class of projective transformations is a superset of a class of affine transformations. By a projective transformation, quadrilaterals transform into quadrilaterals; that is, all the affine motions can be explained by a projective motion. One of the crucial extensions contained in the projective transformation is that the parallel lines need not be preserved.

For feature vectors collected at $t = k$ and $t = k + 1$, we have the following relation:

$$\mathbf{x}_{k+1} = H^T \mathbf{x}_k,$$

where $\mathbf{x}_k = [x_k, y_k, w_k]^T \in \mathbb{R}^3$ is a feature vector at k, and the matrix H represents a 3-by-3 matrix with eight unknown parameters. That is,

$$H = \begin{bmatrix} \theta_1 & \theta_4 & \theta_7 \\ \theta_2 & \theta_5 & \theta_8 \\ \theta_3 & \theta_6 & 1 \end{bmatrix}$$

Those feature vectors are in a homogeneous coordinate system. A popular 2-D transformation is written as:

$$x_{k+1} = \frac{\theta_1 x_k + \theta_2 y_k + \theta_3}{\theta_7 x_k + \theta_8 y_k + 1}, \; y_{k+1} = \frac{\theta_4 x_k + \theta_5 y_k + \theta_6}{\theta_7 x_k + \theta_8 y_k + 1} \tag{18}$$

A plausible drawback of projective registration, instead of affine registration, can be that a computational cost increases when we search an eight-dimensional search space rather than six. Moreover, when the initial parameter is not close to a true solution, the projective transformation is harder to recover; not only because its search space is larger and less constrained, but also because the unknown parameters are highly sensitive to recover. For example, θ_7 is much more sensitive than θ_3 or θ_6. As in Equation (18), the transformation is highly non-linear.

In this part, we revisit the previous affine result [3] with the problem of registering both a Google earth and an UAV image, where the projective map could not be recovered. Now, we want to get the projective result and compare it with the previous one. In
Figure 18, a visual comparison confirms that the projective relation should be considered for these two images. We think that this result can be useful for ortho-rectification purposes, treating the coordinate of the Google earth image as the reference.

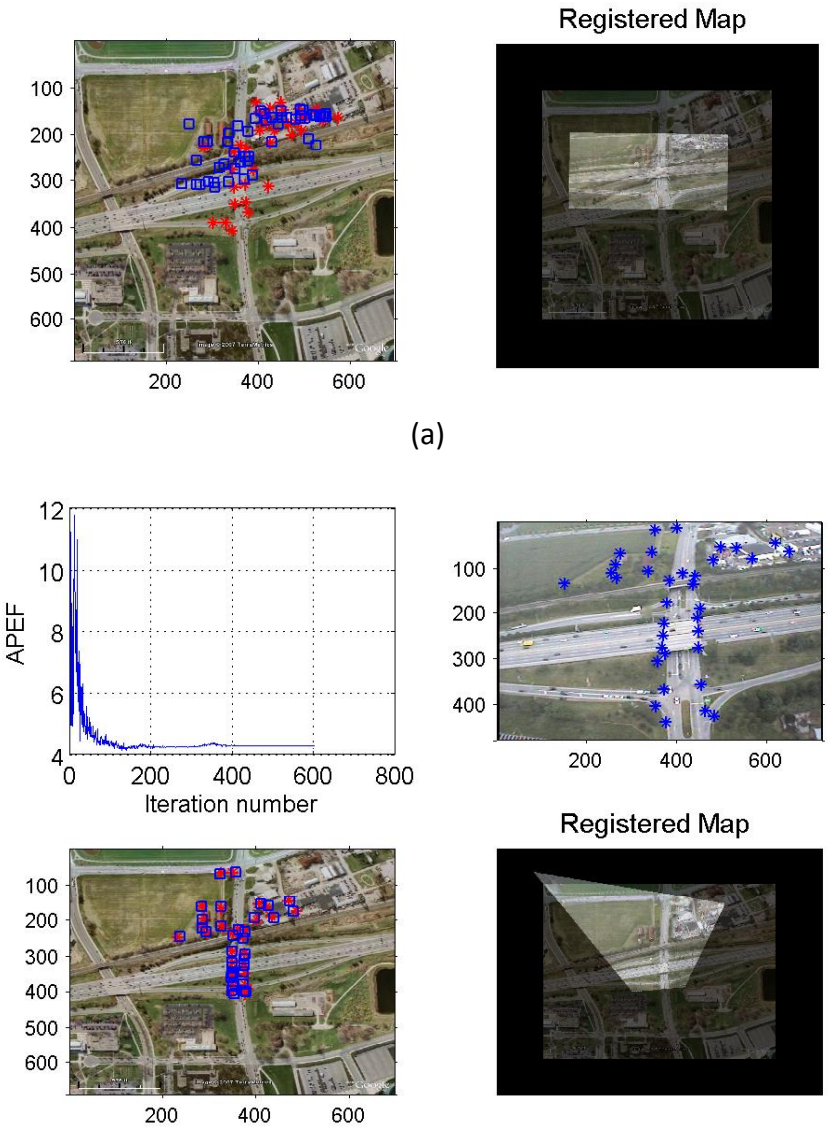


Figure 18. An Example of Data Registration between an Aerial Image from a UAV and a Google Earth Image. The rectified image frame of the aerial image appears on the top of the Google earth image by affine registration in (a) and by projective registration in (b). The projective map, as a result, better explains the relationship between the two images.

5.3.2 Gaussian Scale-space Based Estimation of a Suboptimal Feature Set for RDA

Let $I(\mathbf{x}, t)$ be a video frame over a time space and $I(\mathbf{x}, s, t)$ be an overall data flow over a time-scale space. As mentioned in Section 5.2.1, for a time t, the Gaussian multi-grid representation can generate many versions, $I(\mathbf{x}, s, t)$, of an image at different scales. See Figure 19, where layered sensing based on a UAV creates a 4-D data flow over a time-scale space. The basic idea behind this data structure is that coarse scales can be represented at a lower resolution, while the representation of fine scales requires the full resolution.

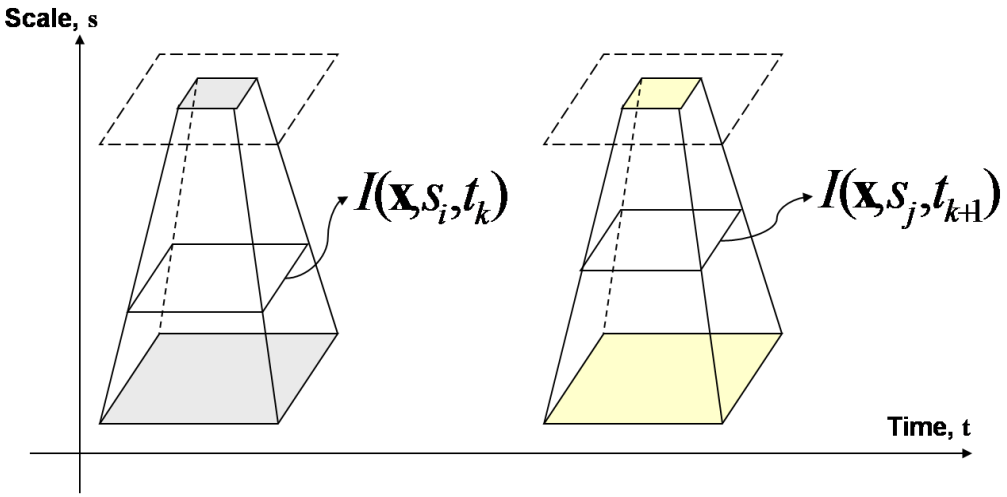


Figure 19. Four Dimensional Data Flow in UAV Sensing

One of the key applications of a differential scale space is a selection of a suboptimal class, at a certain scale, of feature points. The computational cost of our extended RDA algorithm is high, when we have a large number of features. Hence, for an image having a large dimension, we want to reduce the number of feature, say up to less than 50, by using a multi-scale representation of the original high-resolution image. For example, Figure 20 shows that the image size decreases as the scale parameter or the level parameter increases. We also observe that Gaussian smoothing at a higher level provides a lower-resolution image, where the number of features decreases. The fact that some of weak edges are blurred can explain why the number of features in a lower resolution tends to decrease and why the chance of having features on moving vehicles on a highway decreases.

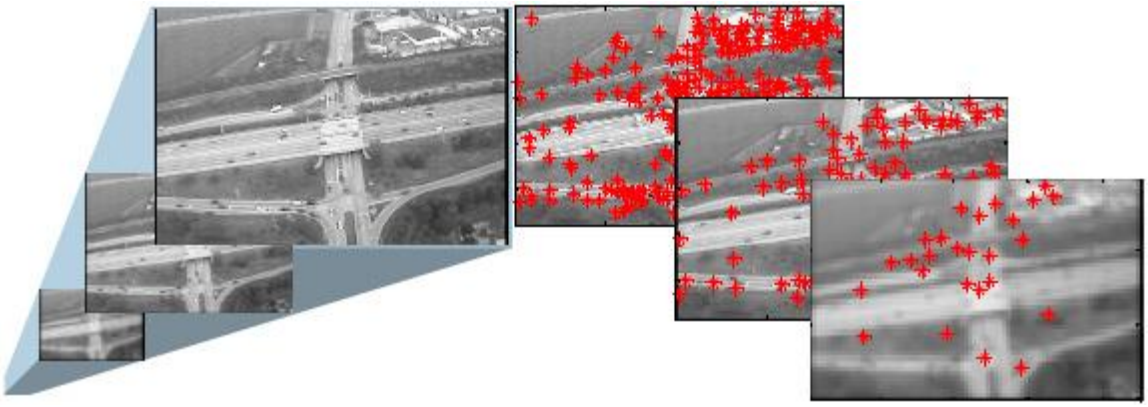


Figure 20. Multi-scale Representation of an Image. Note that possible outliers like vehicles are averaged out at a lower resolution with a smaller image dimension. The number of features decreases in the order of 286, 100, and 29

Figure 21 represents the algorithmic structure of our extended RDA algorithm. The first RDA block can provide an optimal scale for processing features in images. Based on this scale parameter, an approximate version, denoted by $\tilde{L}$, of the higher resolution image, denoted by H, is obtained by $\tilde{L} = \mathcal{B}_{\downarrow 2} H$. Then, the approximate version $\tilde{L}$ is matched again with the lower resolution image, denoted by L.

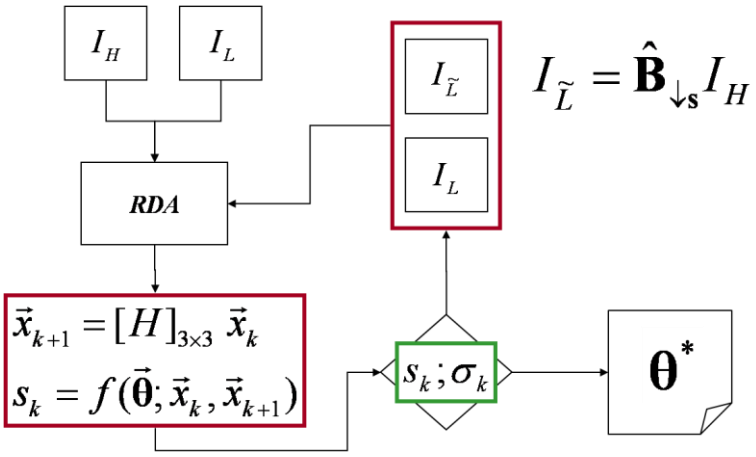


Figure 21. Algorithmic Structure of ERDA

5.4 Experimental Results

5.4.1 Multi-level Registration

Here we apply our extended RDA algorithm to the problem of multi-level data fusion. We believe that there exists an optimal level that provides an optimal number of features. So, we generate the multi-scale representations of image frames and extract features from them based on either the Kanade-Lucas-Tomasi (KLT) [14] or scale-invariant feature transform (SIFT) [27] algorithm in computer vision. Our basic goal is to detect certain features in an image that is optimal at a certain scale.

Given a pair of Electro-optic image frames, we increase each scale parameter by a factor of two and collect four different versions at four different levels for each image. Next, we can collect features automatically; for instance, using the SIFT algorithm, we can extract local features in the four different image versions. Figure 22 illustrates four registration results at levels 4, 3, 2, and 1, from the top left to the bottom right result. We compare the registration at both the level 4 and the level 1 in Figure 23. The registrations at the two different levels are almost identical, so it looks more efficient to start with a higher level rather than with a level 1.

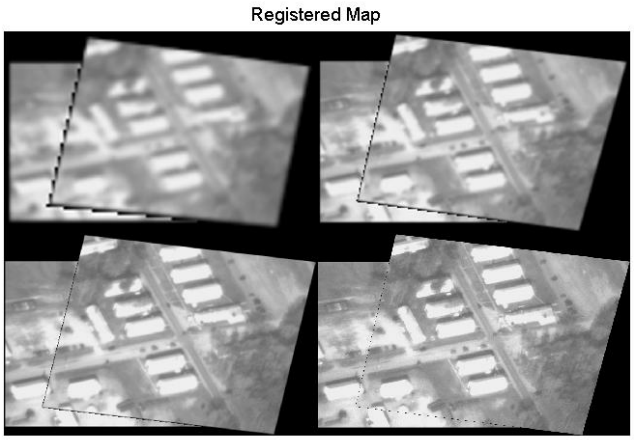


Figure 22. Registered Maps at Four Different Levels. The resolution increases, as the level decreases from the top left to the bottom right output.

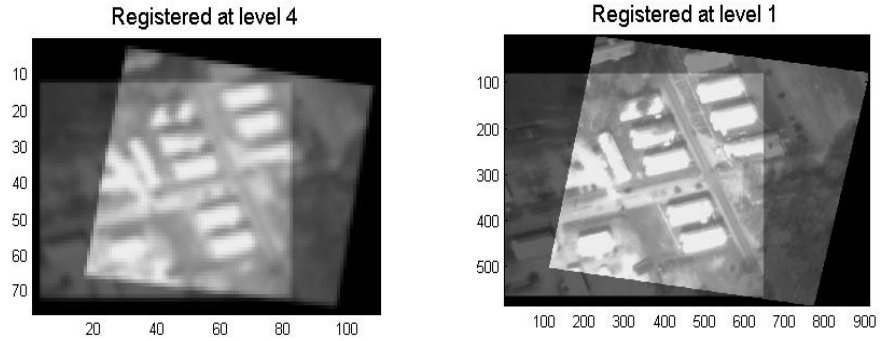
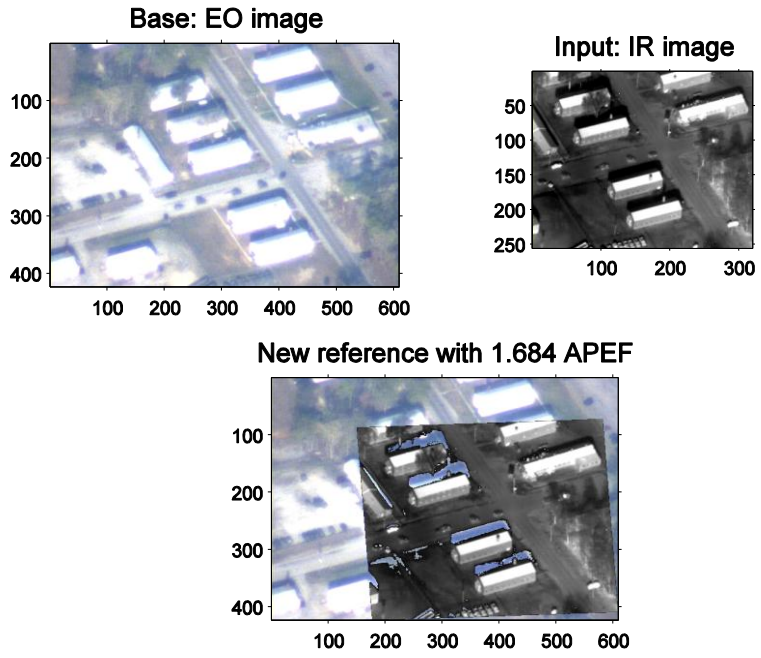


Figure 23. Registration at Two Different Levels

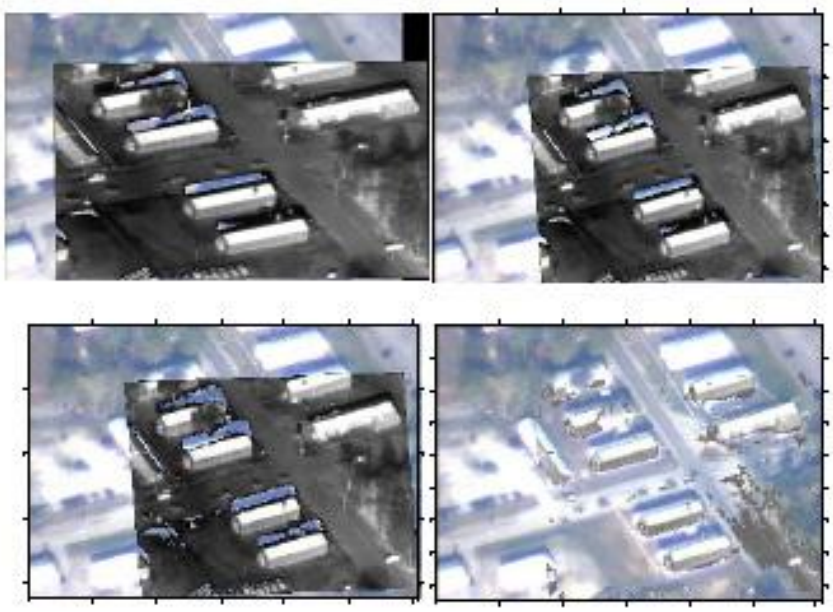
5.4.2 Multimodal Data Registration

Here, we consider the following two different modalities/sensors in DARPA VIVID database: Electro-optical (EO) and infrared (IR) sensors. The EO data represents the intensity of light rays, while the IR data represents the degree of temperature. During daytime hours, the EO and IR sensors can provide related information not only because a common region exists, but also because there exists a connection between the light rays and the heat rays. Here we provide a new experimental result of multi-modal data fusion that successfully combines complementary information from different types of sensors via the process of data refinement in Section 3.4.4.

In Figure 24 by performing multi-modal data registration, the top roofs of multiple buildings in the IR image are superposed on their corresponding parts of the EO image. With a certain threshold, say 100, applied to gray-scaled temperature values, we can use complementary information that the EO image cannot provide. Compared to mono-modal registration performed in either the EO video or the IR video, the relative motions among multiple pairs of EO and IR frames are more sensitive to the choice of an initial parameter vector of an affine map. The automatic selection of the initial parameter vector can be done based on the scores of scaling factors.



(a)



(b)

Figure 24. (a) The IR image is registered into the EO image. (b) Details of data refinement with multiple steps. The bottom right image represents a region above a threshold out of the transferred IR image into the base coordinate

Since our matching algorithm can solve the problem of Multi-modal data registration such as EO and IR video registration, it will give maximal information for further applications using different types of sensors.

In addition, we observe that EO and IR images have a different scaling factor, even though the different modalities come from the two fixed EO and IR sensors attached on the mobile sensor. The recovered transformation reveals one of the fundamental characteristics of the two different modalities.

A possible extension will include multi-modal video registration. In other words, the next step is investigate whether the output of multimodal data alignment given a pair of EO and IR images can help to combine both the IR and the EO videos, and generate a new video sequence that has an additional and complementary layer of other information.

5.4.3 Mosaic Construction and Scene Localization

Here, given a video sequence, we would like to recover a mosaic as a larger field of view. Video mosaics [23] can be useful for scene understanding, persistent tracking, video compression, and enhanced visualization.

In Figure 25, we show a preliminary result of a mosaic from 200 successive IR image frames. While the alignment errors of successive images are small, we notice that those small alignment errors accumulate by composition operations into a reference coordinate. Hence, the result is not in a perfect quality, but still applicable for scenes that are featureless such as forests and straight roads. A new composite mosaic based on RDA can be a more compressed and informative version of a sequence of many images, if we improve this current mosaic quality. In literature [28], a looping path of images has been considered to attack this issue. However, with no loop, it seems still an open problem to find a solution that distributes these accumulated errors in the mosaic. Our future work on video mosaic will include this looping path problem and consider a new constraint: the continuity of an important structure as a road.

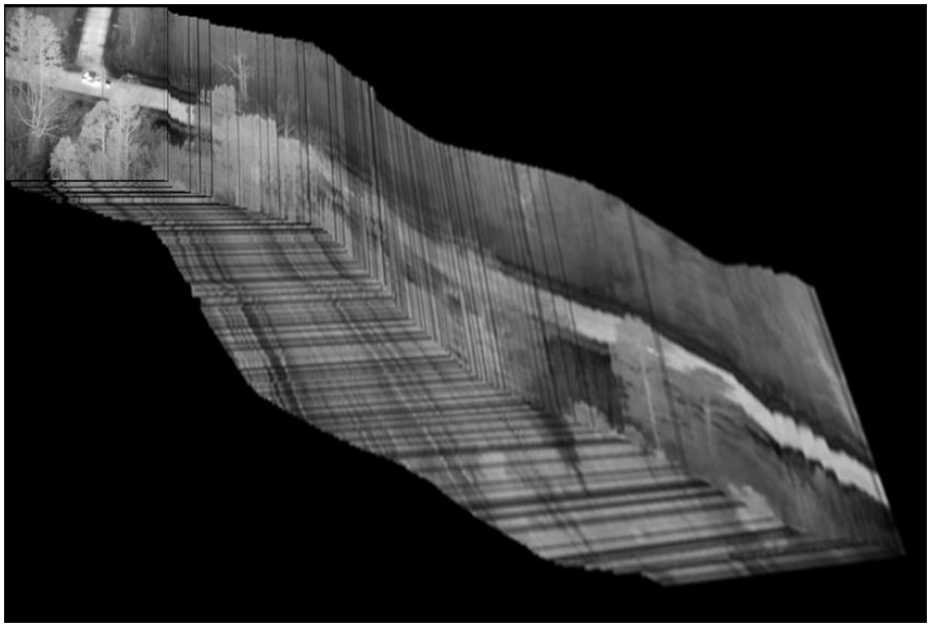


Figure 25. Mosaic from 200 IR Image Frames for Persistent Tracking Applications

Instead of video mosaic, the trajectory of image centers is a simpler way to understand how all the recovered transformations translate image centers into one reference coordinate. In Figure 26, we observe that the centers of 3750 EO image frames reflect a smooth motion of UAV trajectory over a region.

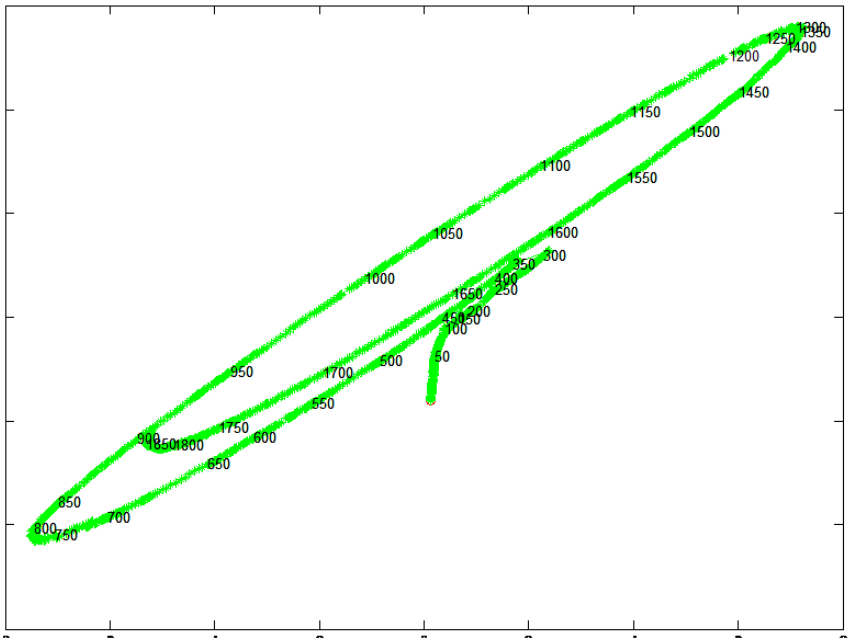


Figure 26. The Trajectory of Image Centers from 3750 EO Images

With the number of small and inexpensive UAVs increasing, it is feasible to build sensing networks for persistent sensing. In this Chapter, we addressed three problems: Multi-level/Multi-resolution data fusion, Multi-modal sensor fusion, and Mosaic construction.

We have extended the previous RDA algorithm to address the three problems. The main result can be summarized as follows:

- Multi-scale and multi-resolution approach can be useful for both scale factor search and an optimal subset of large feature data.
- Information-theoretic RDA algorithm can help combine different modalities such as electro-optic and infrared sensor. Hence, we can register an additional and complementary information layer on top of the base information.
- Mosaic construction and scene localization are still challenging due to an error accumulation, even though our preliminary result can efficiently explain a scene with one image instead of 200 image frames. The trajectory of center images could indicate the looping closing problem for an autonomous navigator.

The investigation of a new local metric is necessary for the current registration algorithm; one of the major issues is how to measure the accuracy of our feature-based and correspondence-less data registration. As a plausible solution, a class of local motion vectors like an optical flow can be chosen over some regions of interest as a subset of the overlapping regions. If it works, the ideal solution

does not provide any motion vectors for a perfect matching. We think that the average norm of these motion vectors can be a good candidate as a new metric for RDA and mosaic quality.

6. Georegistration based on Georeferencing

6.1 Introduction

This report will discuss the problem of registering aerial images. As presented here, the first step in the registration is to use the position and orientation of the camera to warp the images such that the resulting warped image depicts a version of the original which lies along a flat 2D ground map. Pixel locations on the ground plane should be linearly proportional to actual x,y locations in the real world. If the georeferencing of the image is perfect, no further processing is required. The georeferenced image will accurately place its pixels in the real world. However, in general this is not the case, there is often significant error in the position and orientation of cameras. Since the georeferencing is dependent on imperfect information, image to image registration techniques are required to bring the georeferenced images into alignment.

This work will discuss georeferencing, registration without georeferencing, registration with georeferencing on CLIF2007 examples, a two image SIFT example on CLIF2007, and the issues encountered when using georeferencing and SIFT on CLIF 2006 dataset.

6.2 Georeferencing Images

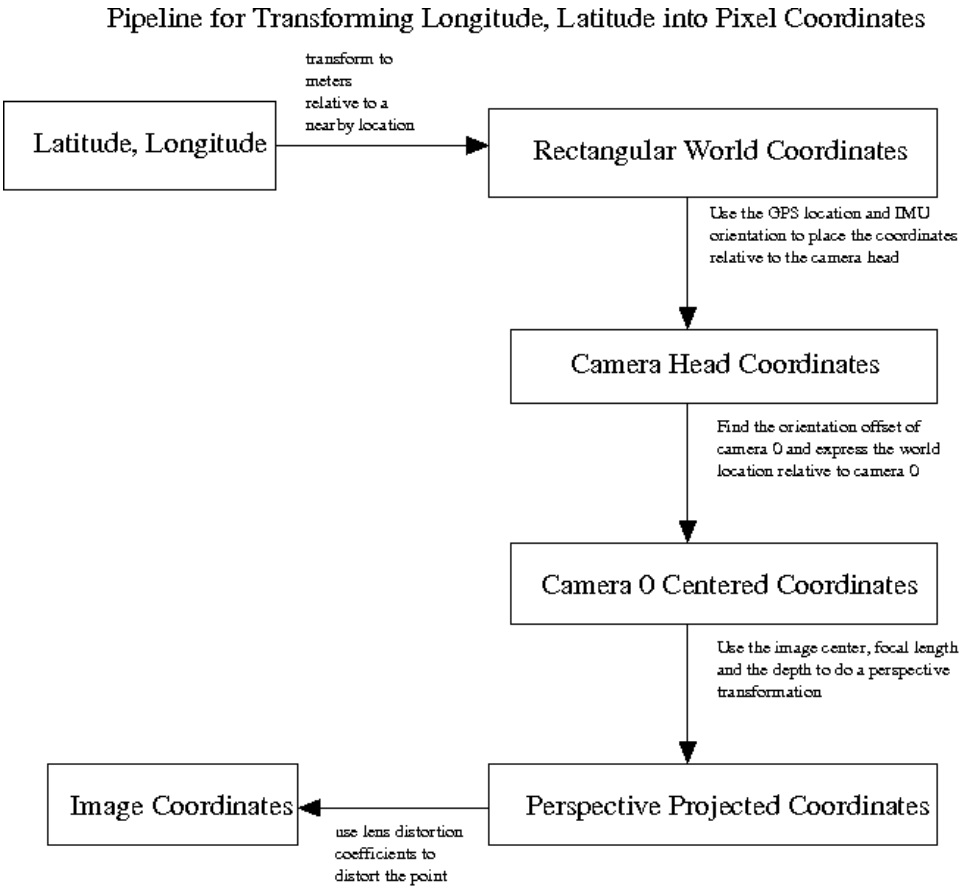


Figure 27. Rendering Pipeline

The georeferencing attempts to find a transform from a point in the world expressed as a longitude, latitude to a corresponding image location.

6.2.1 Longitude, Latitude to Pixel Locations

The process of finding a transform which places a world coordinate correctly within an image is very similar to the rendering pipeline found in 3D computer graphics. Figure 27 shows the rendering pipeline.

First, the longitude and latitude are converted to rectangular world coordinates in meters by using the following expression:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (lon - lon_0)\cos(lat) * 40e^6/360 \\ (lat - lat_0) * 40e^6/360 \\ 0 \end{bmatrix}$$

where $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ is the resulting world location. "z" may be set to zero by assuming the location is on the ground plane. lat_0, lon_0 is an arbitrary nearby location which will be selected as the origin. For the CLIF datasets, this location is in the parking lot for the Center for Automotive Research.

Next, the world coordinates are transformed into camera head coordinates by using the orientation and position of the camera head.

The transformation uses the following matrices:

The pitch matrix:

$$P = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The roll matrix:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\psi) & \sin(\psi) & 0 \\ 0 & -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The yaw matrix:

$$Y = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The translation matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A transformation for making zero degrees yaw lie north along the world's increasing y axis.

$$C_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A transformation for changing the sense of the z axis to point along the camera z.

$$C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The camera head to world transformation is:

$$A\_to\_W = TC_1YPRC_2$$

Similarly, the world to camera head transformation is:

$$W\_to\_A = (TC_1YPRC_2)^{-1}$$

6.2.2 CLIF 2007 Camera 0 Orientation

The cameras in the array are not aligned with the coordinate framework defined by the position and orientation obtained from the IMU. Thus, this orientation offset is part of the gradient descent minimization described below. At present the camera of interest is camera 0 from CLIF 2007. The orientation offset is described by an angle and a vector, where the offset is a rotation of angle α about the vector x, y, z. If this rotation is expressed as a homogenous coordinate transformation matrix C_0 it fits into the above 3D transformations as:

$$W\_to\_C_0 = (TC_1YPRC_2)^{-1}C_0$$

and

$$C_0\_to\_W = [(TC_1YPRC_2)^{-1}C_0]^{-1}$$

6.2.3 Perspective Projection

Once the rectangular world coordinates have been transformed to camera centered coordinates. It is possible to apply a standard perspective projection:

$$x_i = f_x x_w / z_w + c_x$$
$$y_i = f_y y_w / z_w + c_y$$

where x_i, y_i are undistorted image coordinates, x_w, y_w, z_w are the camera centered world coordinates, f_x, f_y is the focal length, and c_x, c_y is the optical image center.

6.2.4 Lens Distortion

The lens distortion model is a standard one found in OpenCV and other sources:

$$x = (x_i - c_x)/f_x$$
$$y = (y_i - c_y)/f_y$$
$$x_d = k_r x + t_x$$
$$y_d = k_r y + t_y$$

where x_d, y_d is the distorted image plane location, and

$$k_r = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6$$
$$r^2 = x^2 + y^2$$
$$t_1 = 2p_1 xy + p_2(r^2 + 2x^2)$$
$$t_2 = p_1(r^2 + 2y^2) + 2p_2 xy$$

In terms of pixels x_d, y_d is

$$x_p = f_x x_d + c_x$$
$$y_p = f_y y_d + c_y$$

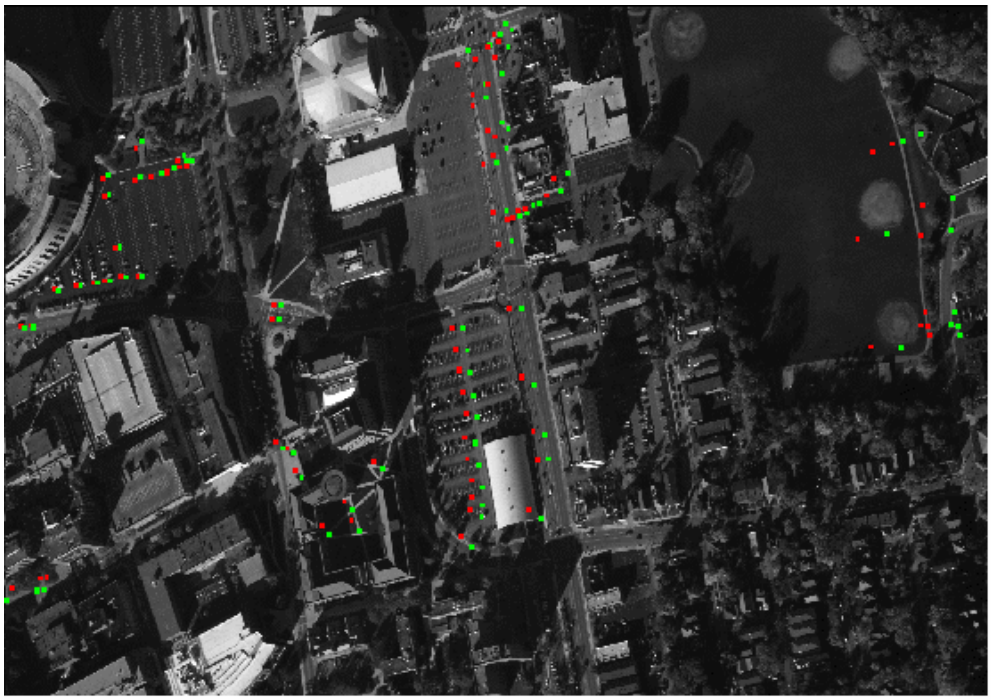


Figure 28. Estimated and Actual Locations

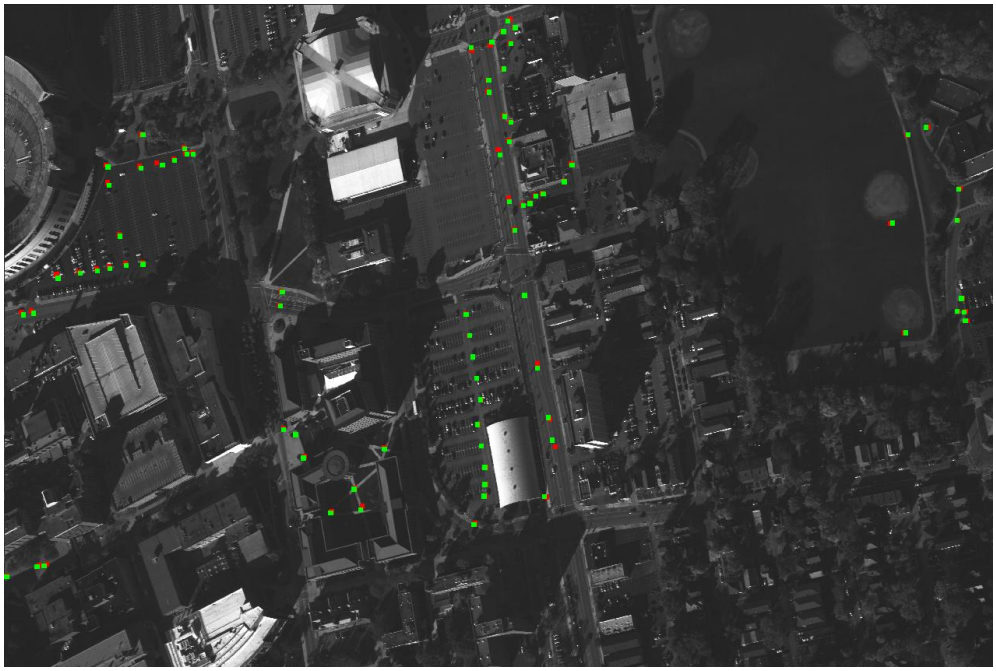


Figure 29. With More Refined Camera Parameters

6.2.5 Finding the Camera Parameters

The ".txt" files in the supplied data do have information about the position and orientation of the camera head such that some of the above matrices can be filled in and evaluated. However, there is not any explicit information concerning the following:

- The orientation of camera 0 with respect to the camera head
- The image center
- The focal length
- The camera lens distortion

The above information can be estimated by minimizing the residuals between the world to image transformation's result and actual image locations of known world positions.

For instance, Figure 28 shows the estimated and actual locations for the series of world/image point pairs. The image in Figure 28 is image 68 camera 0.

Figure 28 displays the estimated and actual locations with the following assumptions

- Camera 0 is rotated 9.10 degrees about the x-axis (roll) of the camera head
- The image center is at 2008,1336 (width/2, height/2)
- The focal length is 15000 pixels
- There is no lens distortion

The green dots are the actual pixel locations corresponding to a longitude and latitude taken from Google Maps. The red dots are the estimated pixel location of a longitude and latitude. One can see that the red and green dots do not coincide. However, they are generally close.

If the residuals between the estimated and actual image locations are minimized using the gradient descent algorithm described below, a more refined camera model results:

- Camera 0 is rotated 9.11 degrees about the vector [0.997 0.068 0.0301]
- The image center is 2066.095,1342.185
- The focal length is 15529.135, 15689.47
- The lens distortion coefficients are

$k_1 = -0.142997, k_2 = 8..782177, k_3 = 0.308878, p_1 = 0.016646, p_2 = 0.020820.$

Figure 29 shows the red and green dots in closer agreement. The more refined camera model may be obtained by minimizing the sum of the residuals between actual and world transformed image locations. In this particular case, this was accomplished with a gradient descent algorithm where the function being minimized is:

$$\min f(f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, \alpha, x, y, z)$$

where the variables are:

f_x, f_y	The focal length in x,y pixels
c_x, c_y	The optical image center
k_1, k_2, k_3, p_1, p_2	The lens distortion coefficients
α, x, y, z	The orientation offset of camera 0 wrt the camera head

The function itself is:

$$f(f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, \alpha, x, y, z) = \sum_i (T(f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, \alpha, x, y, z, w_i) - v_i)$$

where $T(f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, \alpha, x, y, z, w_i)$ is the image location of the world point w_i, given parameters $f_x, f_y, c_x, c_y, k_1, k_2, k_3, p_1, p_2, \alpha, x, y, z$, and v_i is the image point corresponding to w_i. The world point w_i was found by using Google Maps to obtain a longitude and latitude for visually distinctive locations v_i found in the image.

The minimum values are:

f_x	15529.14	Focal length in pixels
f_y	15689.47	Focal length in pixels
c_x	2066.095	Optical image center in pixels
c_y	1342.185	Optical image center in pixels
k_1	-0.142997	Radial lens distortion coefficient
k_2	8.782177	Radial lens distortion coefficient
k_3	-0.308878	Radial lens distortion coefficient
p_1	0.016646	Tangential lens distortion coefficient
p_2	0.020820	Tangential lens distortion coefficient
α	0.158992	3D orientation angle offset in radians about x, y, z
x	0.997232	3D vector element x
y	0.067951	3D vector element y
z	0.030176	3D vector element z

The georeferenced versions of the images found in
Figure 30 and Figure 31 may be found below in Figure 32 and Figure 33.

Figure 30. Sample Image 1: Image Frame 100 from Camera 0

Figure 31. Sample Image 2: Image Frame 110 from Camera 0

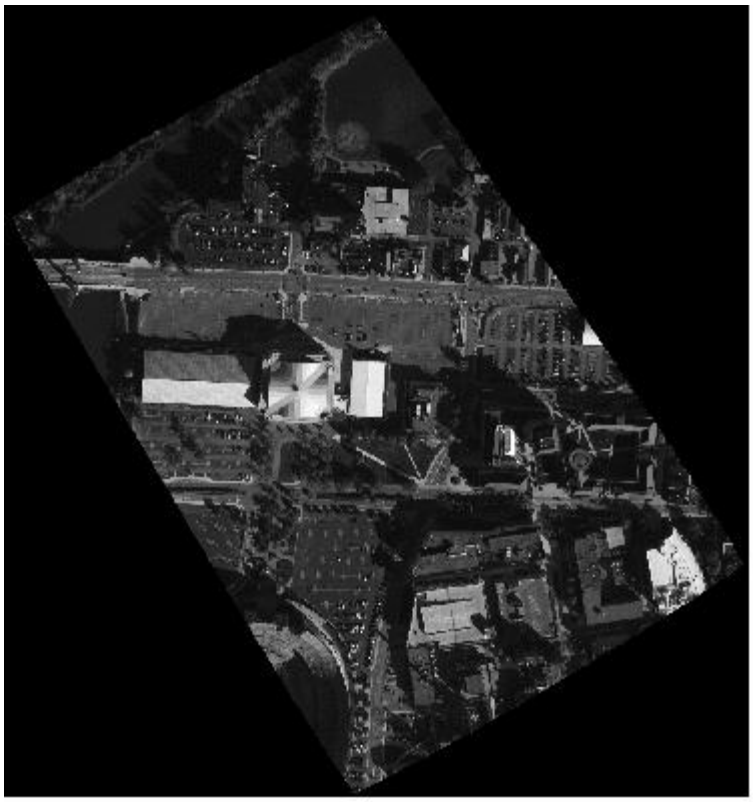


Figure 32. Georeferenced Version of Sample Image 1

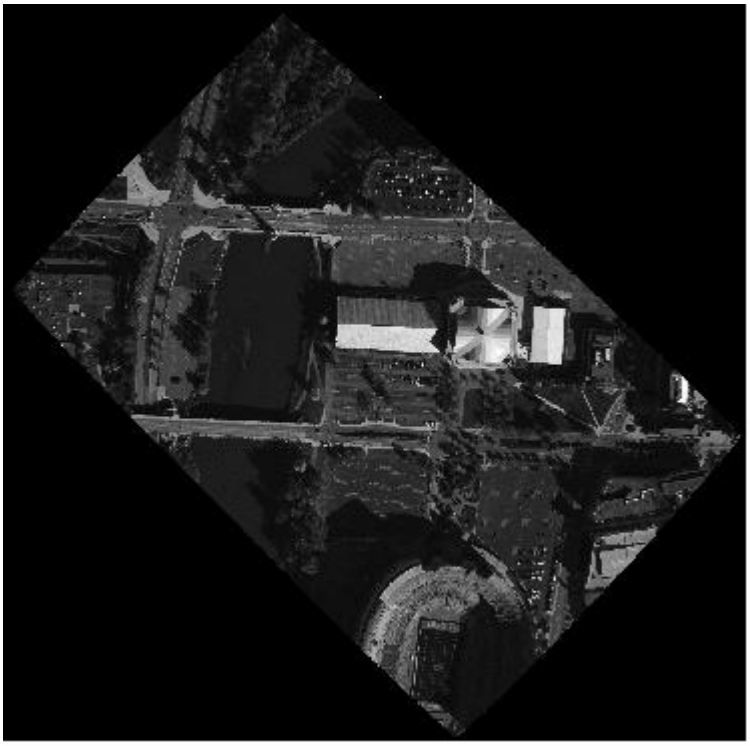


Figure 33. Georeference Version of Sample Image 2

6.3 Image to Image Registration

In addition to georeferencing the images with the position and orientation of the camera, images must be registered by finding a transform from some set of correspondences. Image to image registration attempts to find a transform which can transfer a pixel location in one image to a pixel location in another image. If the pixel locations represent the same point on the same object in the world, this transformation is said to "register" the images.

6.3.1 The Affine Transform

The 2D affine transform may be used to transform locations in one georeferenced image to locations in another georeferenced image.

A two dimensional affine transform is given by:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

where $\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$ is the transformed location of $\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$.

Although the 2D affine transform can rotate, scale, shear and translate two-dimensional data, a 2D affine transform has two major problems when used to register general non-georeferenced images. First, while a 2D affine transform can represent an orthographic projection of a 3D object, it cannot generally represent a perspective projection of a 3D object. This means that images of objects which are not perpendicular to the z-axis of the imaging camera cannot be registered exactly with a 2D affine transform. Second, any camera that uses a lens suffers from lens distortion that cannot generally be modeled with a 2D affine transform.

6.3.2 Using Correspondences to find an Affine Transform

A registering affine transform may be found from a set of correspondences. If the set of correspondences does not have any wild outliers, a straightforward least squares formulation may be used. One may find the six 2D affine transform parameters x such that $|Ax - b|$ is minimized where

$$x = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \\ t_x \\ t_y \end{bmatrix}$$

and A is a $2n$ by 6 matrix where n is the number of correspondences and A is:

$$A = \begin{bmatrix} x_{1i} & y_{1i} & 0 & 0 & 1 & 0 \\ 0 & 0 & x_{1i} & y_{1i} & 0 & 1 \end{bmatrix}$$

and B is a vector of size $2n$:

$$B = \begin{bmatrix} x_{2i} \\ y_{2i} \end{bmatrix}$$

and i indexes the correspondences x_{1i}, y_{1i} to x_{2i}, y_{2i}.

Generally, however, the set of correspondences have significant outliers which need to be removed before further evaluation. For this reason, it is necessary to include a Random Sample Consensus (RANSAC) initial preprocessing step.

6.3.3 RANSAC Preprocessing

RANSAC is a general approach to fitting a primitive such as a 2D line or 2D affine transform to a set of data. The case of fitting a line to a single set of 2D points is particularly simple and will serve as an example. In this case, two random points are selected, and the line defined by these two random points is found. This line is a hypothesis for the desired result, and this hypothesis line may be tested for fit against the existing set of 2D points. After many hypothesis line trials, the hypothesis line which has the best support in the 2D point set is the desired result.

The primary advantage of a RANSAC approach over a least squares fit is that RANSAC is capable of completely ignoring outliers. While a least squares fit may be significantly affected by outliers, a RANSAC fit will focus on a single primitive interpretation of the data. The primary disadvantage of a RANSAC approach is that it makes an incorrect assumption that the data points which are selected for the winning hypothesis formulation are perfect and error-free. Given the advantages and disadvantages of RANSAC, this registration algorithm uses a combination of RANSAC and least squares. First, outliers are eliminated with RANSAC. Second, the remaining points are used in least squares to find the final transform.

The 2D affine transform RANSAC is different in several details from the line-finding RANSAC example described above. Rather than selecting two random data points from a single set of data, the 2D affine transform RANSAC selects three correspondences from two data sets, and solves the now fully determined equations described above in the least squares discussion with 6 equations and 6 unknowns. Pseudo code for the 2D affine RANSAC follows:

```
largest_number_of_support_points=0
for each trial
    corrs=select_three_correspondences
    affine_hypothesis=solve_for_affine_xform(corrs)
    number_of_support_points=get_number_of_support_points(affine_hypothesis)
    if (number_of_support_points>largest_number_of_support_points)
            best_affine_hypothesis=affine_hypothesis
            largest_number_of_support_points=number_of_support_points
```

6.4 Finding Correspondences

Several methods of finding correspondences were tried during this study. Some methods were more successful than others, and the method that seems to offer the highest performance in the sense of being robust against large changes in lighting, position changes and orientation changes is the Scale Invariant Feature Transform(SIFT) algorithm developed by David Lowe. However, initially, much of this work was accomplished by using the algorithm described below.

6.4.1 Point Pattern Matching

Figure 34 show two overlapping images. In each image there are red dots representing the locations of corners. In this case the corners were found with OpenCV's KLT corner detector. An affine transform is desired which minimizes some cost function of the registration error between each image's sets of corners.

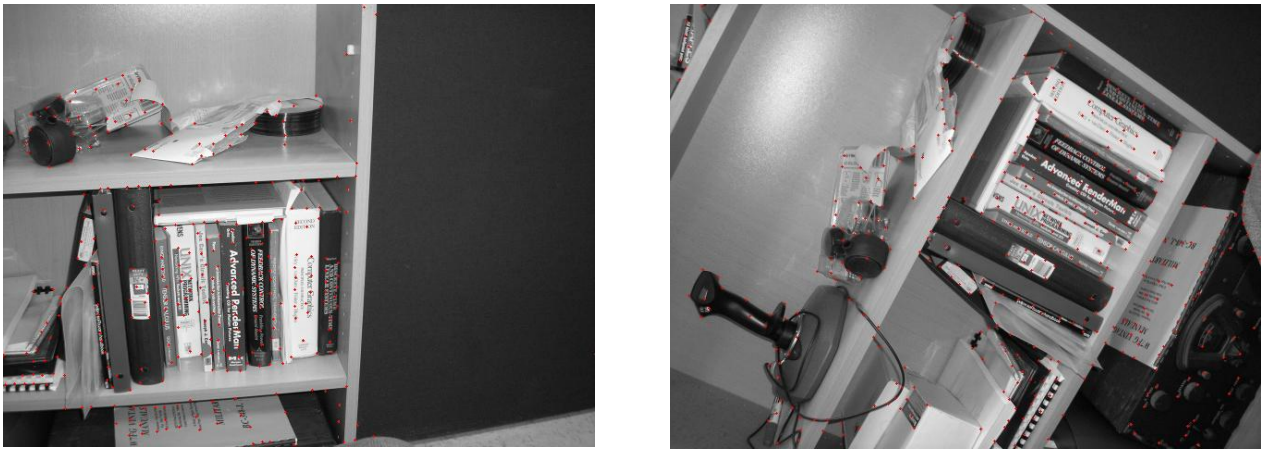


Figure 34. Corner Features in Two Overlapping Images

In this approach a RANSAC type algorithm is used to identify a set of correspondences that generate an affine transform minimizing the cost function. The discovered correspondences are displayed in Figure 35 has red lines drawn between image locations.

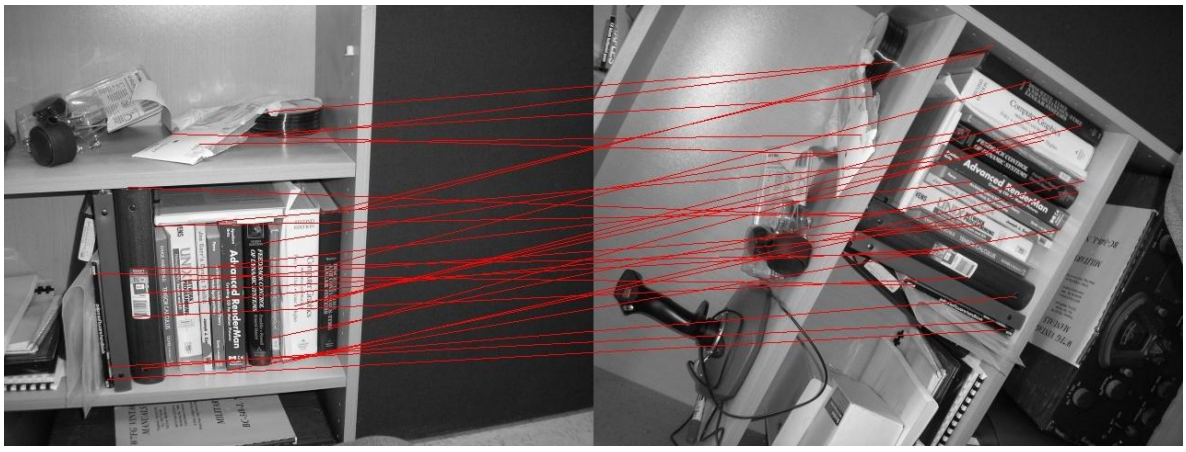


Figure 35. Discovered Correspondence

The algorithm first does a point by point comparison between the two sets of corners. Unlike many algorithms, it does not use a rotation and noise invariant feature description vector to decide whether two corners are likely to represent the same location on the same object. Rather, the algorithm will look at the neighbors of the two points under consideration and attempt to find a rigid 2D affine transform which will make the neighbors match according to some cost function.

As an example, Figure 36 shows two points, P1 and P2, being compared with each other. In this case, P1 and P2 are judged to be very similar because a rigid 2D affine transform can be found which will make the neighbors of P1 and P2 match. This matching rigid 2D affine transform is found by evaluating RANSAC hypotheses. These RANSAC hypotheses are generated by randomly selecting one neighboring point correspondence in addition to the center point P1 to P2 correspondence. The use of the rigid 2D affine transform rather than the general 2D affine transform allows the use of two correspondences rather than the usual three.

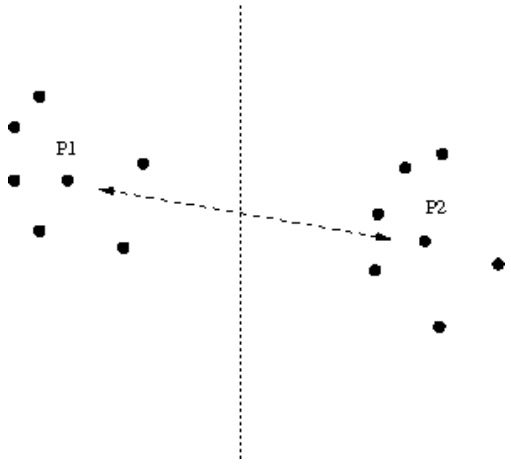


Figure 36. RANSAC Hypotheses

If a rigid 2D affine transform cannot be found which matches P1 and P2 and their neighbors, then P1 and P2 are judged to be unlikely correspondences, and another two points will be tested.

After a set of potentially matching points is found, a set of non-rigid 2D affine transform RANSAC hypotheses are generated which are tested according to the global cost function. The best RANSAC hypothesis is retained and used to cull from the correspondence set any high residual correspondences. The remaining correspondences are then dropped into a least squares minimization function and a resulting 2D affine transform is returned as the result.

6.4.2 Aerial Imagery

Figure 30 and Figure 31 show images 100 and 110 from camera 0 CLIF 2007 dataset. The registration algorithm attempts to find a 2D affine transform which best fits the corners.

Figure 37 shows the correspondence chips found by the algorithm. Because the aerial images are so much larger in terms of resolution than the desk scene shown in

Figure 34, the thresholds on the residuals had to be increased in order to accept the 24 correspondences shown in Figure 37. These 24 correspondences resulted in the following affine transform:

$$R = \begin{bmatrix} 0.932634 & 0.242830 \\ -0.288799 & 0.937685 \end{bmatrix}$$

$$T = \begin{bmatrix} -592.47 \\ 1733.17 \end{bmatrix}$$

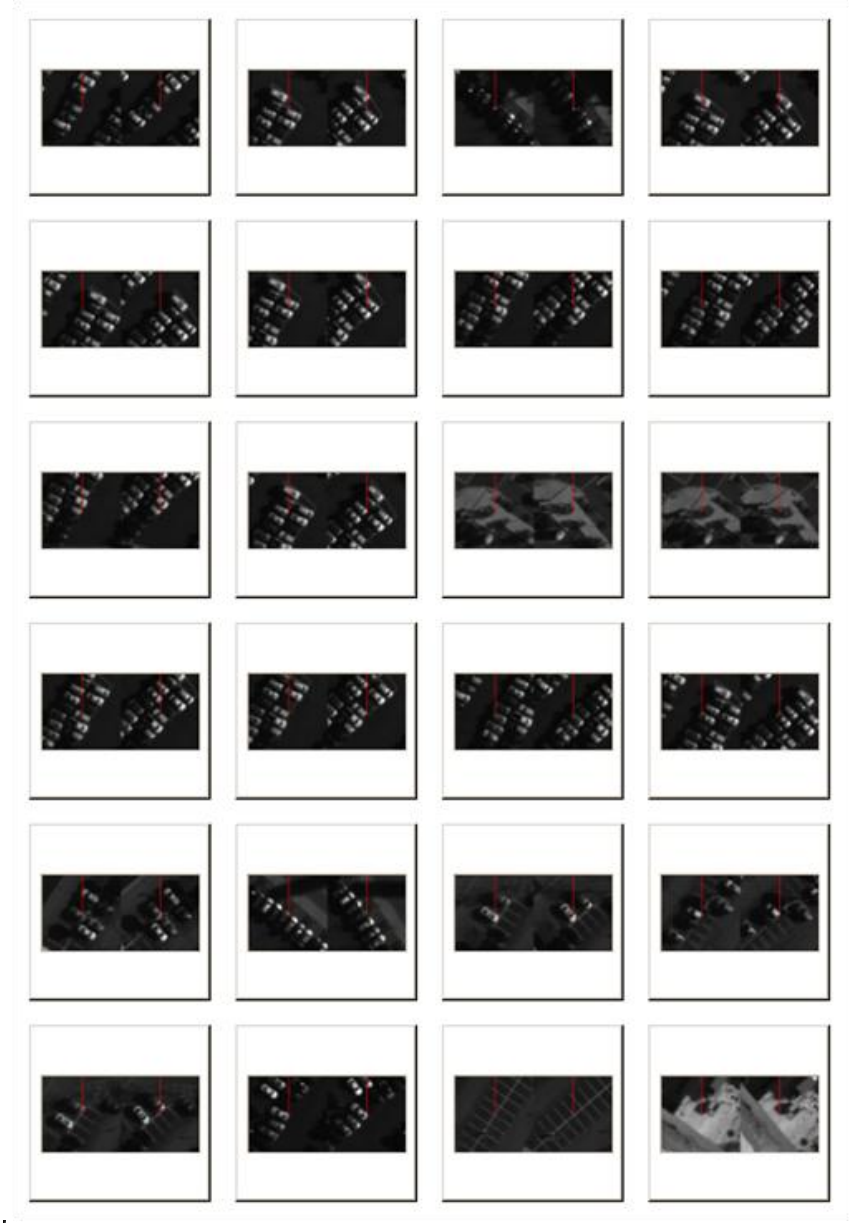


Figure 37. 24 Correspondences

6.4.3 Error in the Non-Georeferenced Example

Particularly with large images, it's possible for the resulting 2D affine transform to have areas of the image where pixels have an unacceptable error. If one is attempting to track vehicles, this

error can be a significant fraction of a vehicle length. The average error for the randomly selected set of points is 4.58 pixels. Any random image locations which were on rooftops were excluded. Also, any image location set which didn't show a distinct error offset was excluded. Finally, any image location set which transformed to a non-image location was excluded.

Chip Name	X pixels	Y pixels	pixel error
775_1485_to_1421_173.ppm	7	5	8.602325267
670_2512_to_1053_1155.ppm	12	11	16.2788206
2766_1990_to_3268_1280.ppm	0	4	4
2694_1336_to_3364_613.ppm	-3	-8	8.544003745
2204_2434_to_2596_1546.ppm	0	4	4
2195_2259_to_2631_1372.ppm	-2	-2	2.828427125
2190_1886_to_2723_1001.ppm	-6	-1	6.08276253
2056_2008_to_2558_1081.ppm	0	0	0
2012_2265_to_2448_1321.ppm	-2	-4	4.472135955
1932_2557_to_2294_1586.ppm	0	0	0
1846_2017_to_2347_1026.ppm	-3	-1	3.16227766
1606_2514_to_1982_1443.ppm	3	2	3.605551275
1501_1285_to_2193_198.ppm	-1	-3	3.16227766
985_2083_to_1476_828.ppm	7	2	7.280109889
1260_1396_to_1925_234.ppm	0	0	0
1242_1716_to_1825_544.ppm	0	0	0
1078_1809_to_1638_585.ppm	5	3	5.830951895
		total:	77.8496436
		avg:	4.5793908

Table 2. Computed Errors in Non-Georeferenced Images

6.4.4 Error with the Georeferenced CLIF2007 Images 100 110

Theoretically, the rectified images are already registered. However, technically, this is not the case. Small orientation errors in the IMU data will cause the same world location to be offset from image to image. However, if the registration algorithm is now applied to these images, the 2D affine transform may be more applicable since the images, except for tall buildings, lie "flat". The rectified images in Figure 38 have been translated such that the black regions around the periphery are minimized. Note that the corners of the rectified image contact the edges of the image. Thus, the translation offset provided by the transform to world coordinates is not nearly as large as seen in Figure 32 and Figure 33.

If the affine transform for Figure 32 and Figure 33 is found by the algorithm described above, the following transform results:

$$R = \begin{bmatrix} 0.992141 & -0.000093 \\ -0.001619 & 0.998812 \end{bmatrix}$$

$$T = \begin{bmatrix} -955.48 \\ 250.58 \end{bmatrix}$$

Note that the rotation matrix is very nearly the identity matrix, while the translation portion reflects the translations necessary to get the rectified images to fit into a small rectangular frame.

Chip Name	X pixels	Y pixels	pixel error
970_2428_to_1942_2186.ppm	-3	0	3
945_1407_to_1915_1162.ppm	-2	0	2
815_1460_to_1786_1215.ppm	-1	1	1.414213562
769_1322_to_1740_1077.ppm	-1	-1	1.414213562
689_1376_to_1660_1131.ppm	-1	0	1
677_1460_to_1648_1215.ppm	-1	0	1
583_2012_to_1556_1768.ppm	-1	-3	3.16227766
564_1402_to_1535_1157.ppm	0	0	0
1898_2343_to_2866_2103.ppm	5	1	5.099019514
1888_2329_to_2856_2089.ppm	4	2	4.472135955
1715_3184_to_2686_2946.ppm	2	0	2
1703_2738_to_2673_2498.ppm	3	0	3
1679_3061_to_2650_2822.ppm	2	0	2
1616_2917_to_2587_2677.ppm	0	0	0
1602_1598_to_2570_1356.ppm	2	0	2
1530_1808_to_2499_1566.ppm	0	0	0
1467_2098_to_2437_1857.ppm	0	0	0
		total:	31.56186025
		avg:	1.856580015

Table 3. Computed Errors in Georeferenced Images

6.4.5 SIFT

It was found that the SIFT algorithm outperforms the point pattern matching algorithm described above. In the latter stages of this work, the focus moved to the SIFT algorithm.

The SIFT algorithm operates by first finding feature points. It creates a set of difference of Gaussian (DOG) images at steadily increasing scale. The DOG images are then searched for local extrema. These local extrema form SIFT features. The x, y location and scale of each SIFT feature is noted, and a feature orientation is found by locating maximum bins in a gradient histogram. If there is more than one dominant direction, the feature is divided into two different features.

The DOG images are intended to be an approximation to the Laplacian of Gaussians scale space operator which attempts to find maxima of the divergence of the image gradient as the image is convolved with Gaussian functions of increasing sigma.

Once the SIFT features are found, the location, the scale and the orientation of the feature is used to find a 128 item descriptor vector for the feature. The SIFT feature descriptors are matched by using a kd-tree to efficiently find Cartesian nearest neighbors. A kd-tree places a set points either to the right or left of a root node depending on whether a selected vector coordinate is greater than or less than the root node's value. A kd-tree is similar to the binary space partitioning (BSP) tree used in polygon rendering except that the dividing planes in a BSP-tree are arbitrary while a kd-tree's planes are perpendicular to the axes of the space generated by the vector-valued points which are placed in the tree.

6.4.6 Two-Image SIFT Example

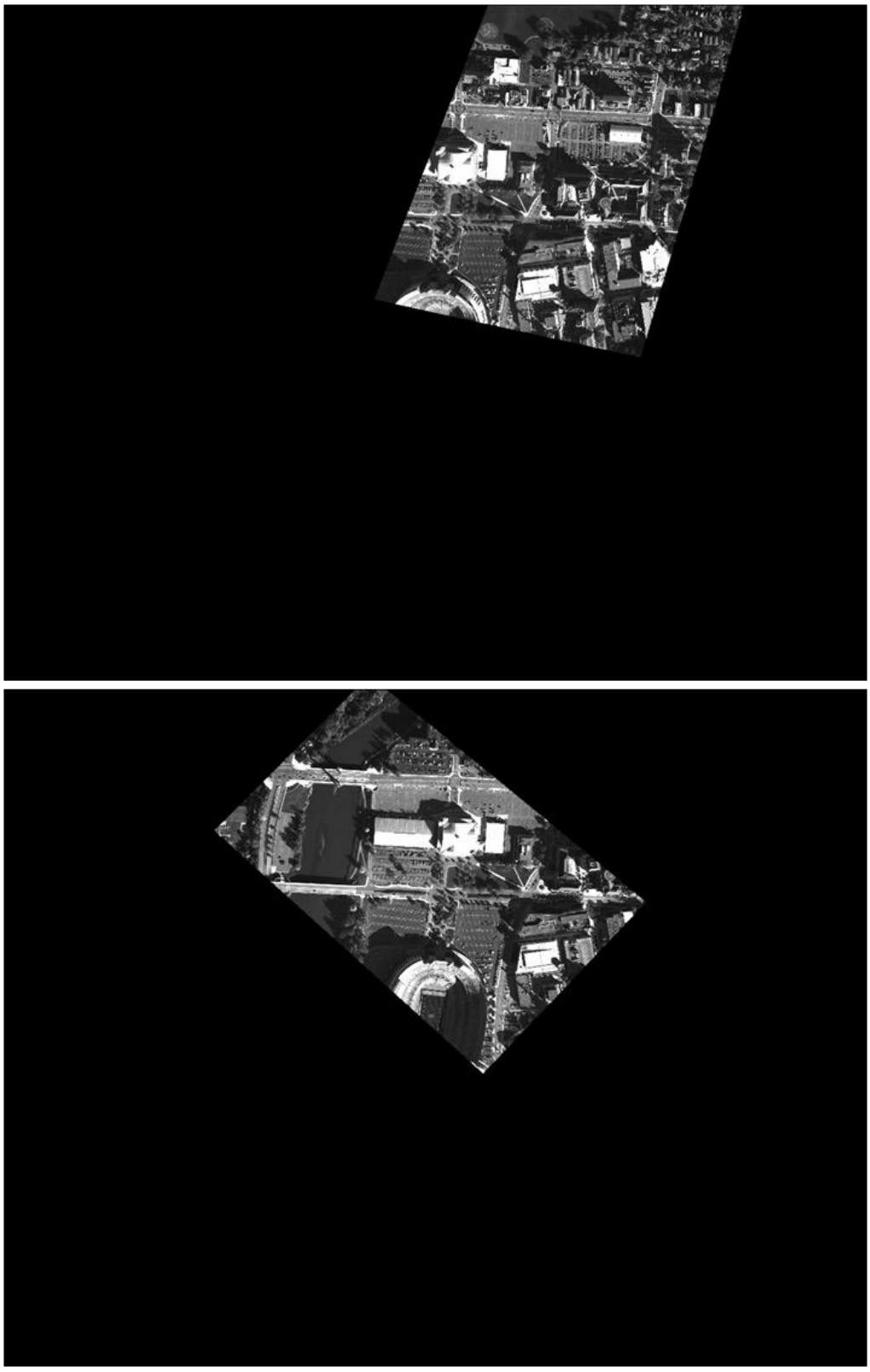


Figure 38. Two Rectified Aerial Images: Frame 68 and 568 from Camera 0

As an example of finding SIFT correspondences in aerial images, consider Figure 38 where two georeferenced images centered on Ohio Stadium are 500 frames apart taken from the CLIF 2007 dataset. Specifically, they are camera 0 frame 68 and camera 0 frame 568.

In Figure 38, the stadium is visible at the center of the 1.8 x 1.5 km frame, with St. John's arena just to the north of the stadium. These images have been artificially lightened since the camera 0 CLIF 2007 data tends to be dark and low contrast. The images have significant overlap where correspondences may be found. Several example correspondences out of 161 are shown below in Figure 39, these correspondences are taken from the raw image, so the contrast is not particularly good.

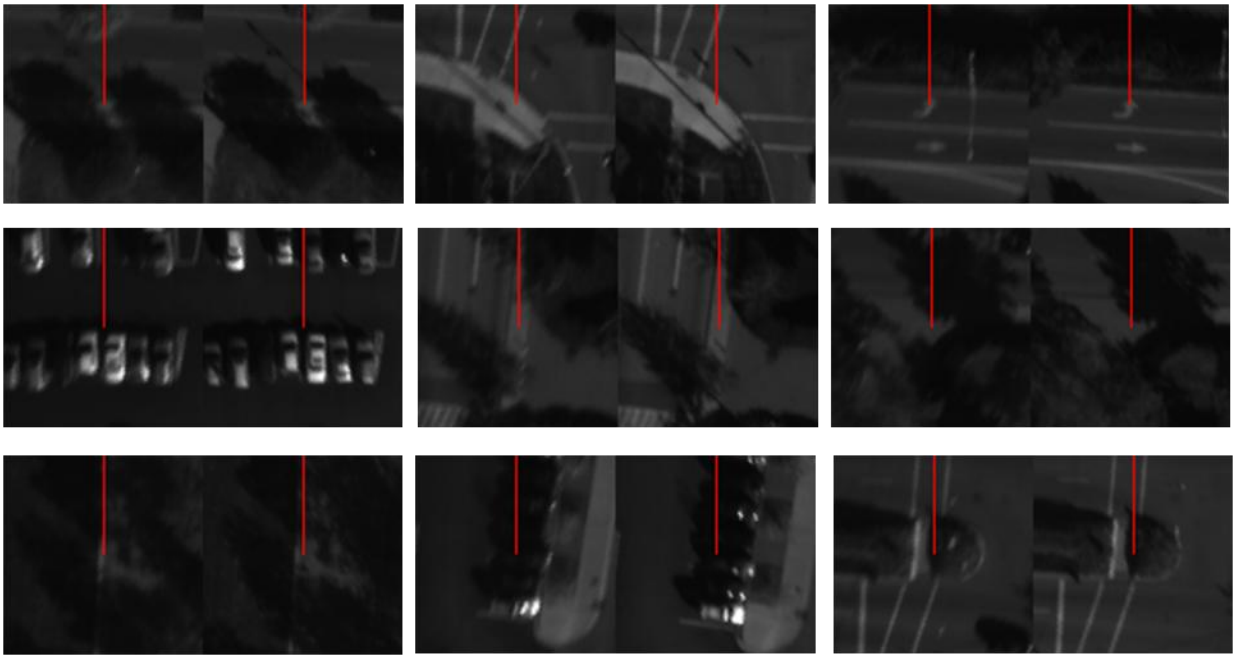


Figure 39. Examples of Correspondences

6.4.7 CLIF2007 and Electro-Optical Mid-Level Images

A similar example of using SIFT was also done using CLIF 2007 images and the mid-level EO data, and the results for finding correspondences were similarly good even though the images were taken by different cameras at different times. However, the mid-level EO images are inconvenient to include in this report due to ITAR restrictions.

6.5 Georeferencing the CLIF 2006 Data

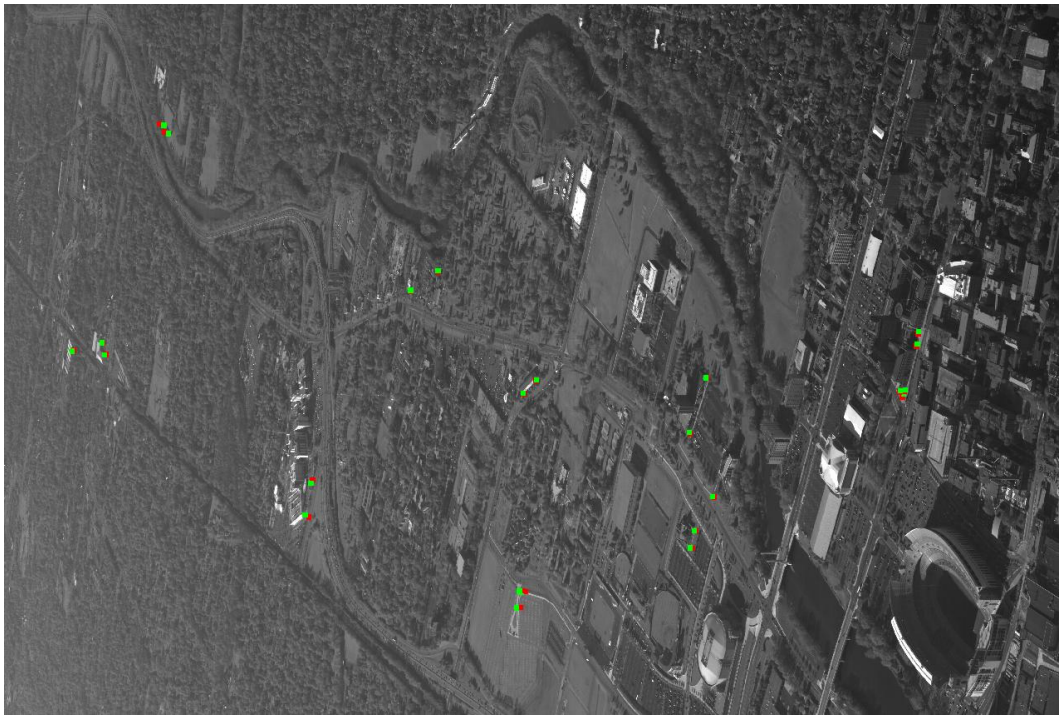


Figure 40. Frame 000003-003051 from CLIF 2006 Dataset

Figure 40 shows the results of the calibration procedure for frame 000003-003051 from the CLIF 2006 dataset. The green dots in Figure 40 are image locations while the red dots are transformed longitude and latitudes. Expressed as a table, the above data looks like Table 5 where there are 26 widely spaced correspondences between the green image coordinates and the red longitude and latitudes. The minimized residual sum of 248.7 pixels gives an average error of 9.6 pixels. The minimizing camera parameters are summarized Table 4.

Rotation Angle	-8.09 degrees
Rotation Vector X	0.998639
Rotation Vector Y	0.041411
Rotation Vector Z	0.031710
Focal Length X	8838.48
Focal Length Y	9488.43
Optical Center X	2005.24
Optical Center Y	1299.68
Distortion Coef k1	0.032679
Distortion Coef k2	2.744133
Distortion Coef k3	0.165224
Distortion Coef p1	0.008055
Distortion Coef p2	0.011938

Table 4. Camera Parameters

X image (green)	Y image (green)	Latitude (red)	Longitude (red)
3416	1464	40.004254	-83.017049
3423	1448	40.004236	-83.016922
3403	1452	40.004369	-83.017015
3468	1277	40.004211	-83.015972
3474	1230	40.004239	-83.015734
2602	1609	40.010473	-83.021592
2666	1403	40.010284	-83.020114
2691	1850	40.009179	-83.022335
2606	2042	40.009571	-83.023856
2620	1978	40.009576	-83.023427
2022	1411	40.017301	-83.024223
1971	1461	40.017804	-83.024894
1647	1000	40.023578	-83.024451
1543	1074	40.025100	-83.025830
1142	1918	40.029804	-83.036328
1164	1799	40.029919	-83.035057
381	1317	40.053234	-83.043621
370	1272	40.053936	-83.043466
257	1303	40.058116	-83.046134
623	485	40.049002	-83.031398
606	454	40.049686	-83.031350
623	485	40.049002	-83.031398
606	454	40.049686	-83.031350
1946	2268	40.016097	-83.029982
1964	2208	40.015995	-83.029436
1957	2205	40.016116	-83.029497

Table 5. Image to World Correspondences

If one uses the camera parameters found in Table 4 to warp image 000003-003051 (the original in Figure 27) such that the rectangular coordinates derived from the longitude and latitude lie in an image plane centered on Ohio Stadium (40.001616,-83.019733). The image in Figure 41 results. Figure 28 represents an approximately 1.8 x 1.5 kilometer region (9008 x 7672 pixels at 0.2 meters/pixel).

The warped image is not entirely contained in the 1.8 x 1.5 kilometer output region because portions of the image are simply too far away. The CLIF2006 dataset uses a shorter focal length lens than the CLIF2007 dataset, and the left side of the image in Figure 40 is rather close to the horizon. In the particular case of the warped image in Figure 41, the left side of the image is 5.4 kilometers away from the center of Ohio Stadium.

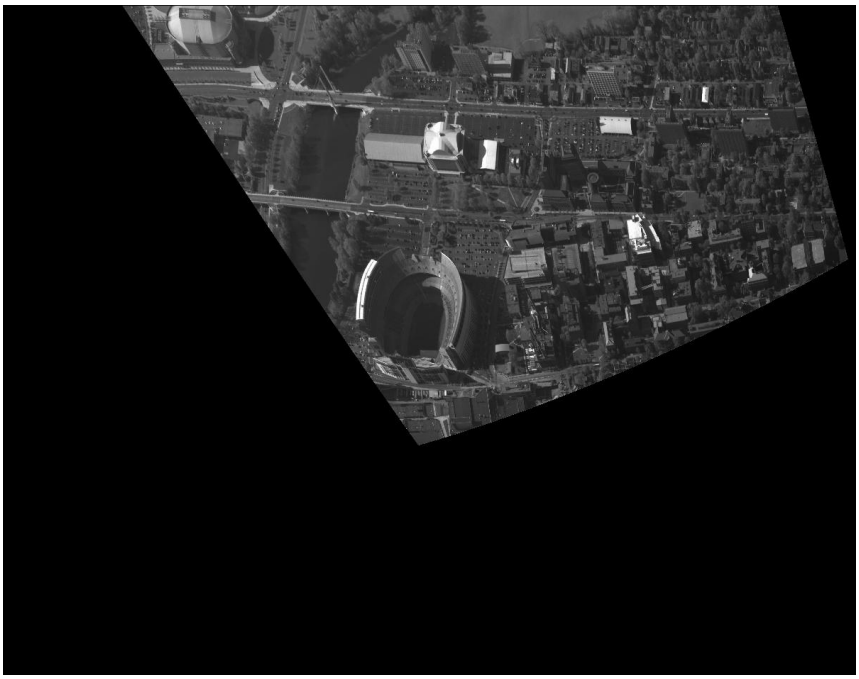


Figure 41. Warped 000003-003053

A sequence of georeferenced frames similar to three examples (000003-002200,000003-002250 and 000003-002300) seen in Figure 42 can be created from the camera parameters found in Table 2. Ideally, if the imu, gps and calibration are perfect, these warped images will be stabilized as well. However, the IMU is far from error free. The resulting georeferenced images have significant amounts of error.

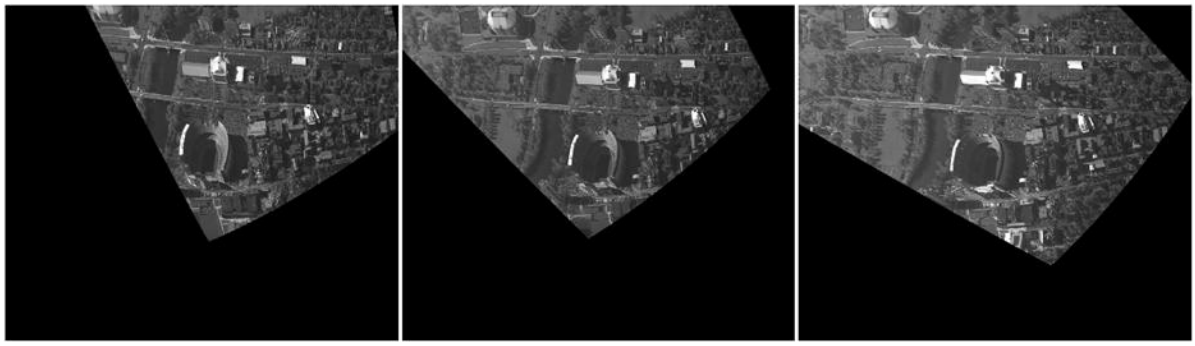


Figure 42. Georeferenced Frames with Errors

6.5.1 Errors from IMU, Calibration in CLIF 2006

Due to the IMU/calibration error, georeferencing the images with IMU data is insufficient for stabilization. However, a set of SIFT correspondences may be found which can provide an affine transform which attempts to bring the georeferenced images into alignment. Because the georeferenced images are supposed to be orthonormal, the affine transform is more appropriate than a homography or projective transform. Generally, the IMU derived georeferenced images do not require a great deal of scaling or rotation, so the 2x2 affine matrix tends to be close to the

identity matrix. However, the translation required to bring the georeferenced images into alignment can be substantial. This translation can be recorded and reported as a reasonable metric for IMU/calibration error. This error is shown in Figure 43.

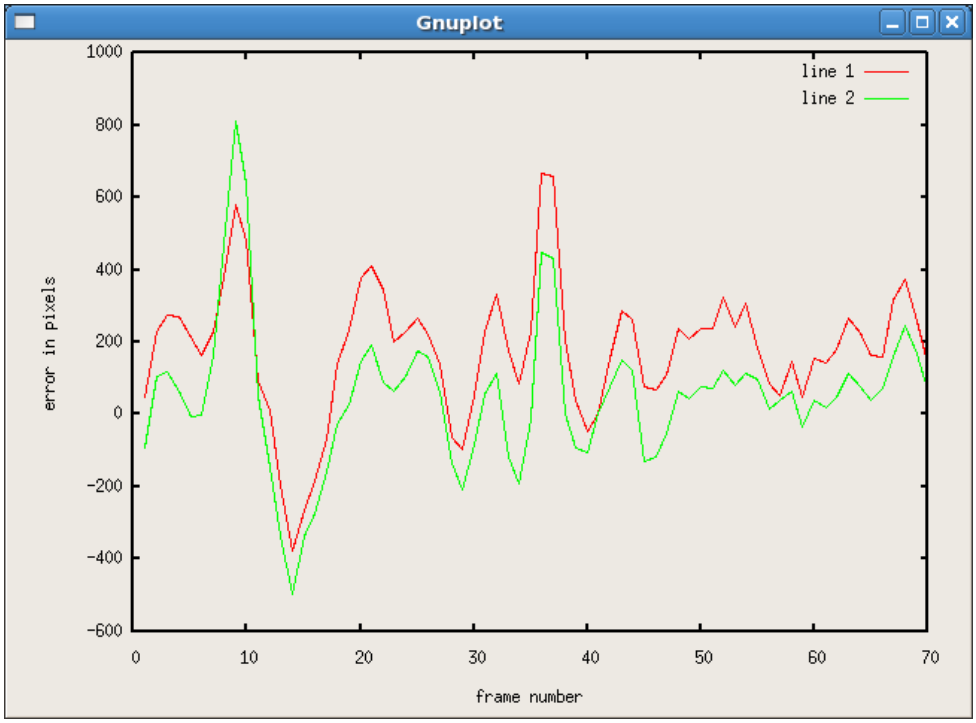


Figure 43. SIFT Alignment Translation

Figure 43 shows the alignment translations for 70 frames from 000003-002230 to 000003-002300. The green line represents the x error while the red line represents the y. Figure 44 shows the total error. The error is substantial. The average total error is 270 pixels. Since the images are georeferenced to 0.2 meters per pixel, 270 pixels represents 54 meters. This error may be large enough that the assumption that the perspective effects of the camera have been eliminated or ameliorated by the georeferencing may be false. The error is also high enough that the elevation for a particular pixel cannot be found from a DEM with sufficient accuracy.

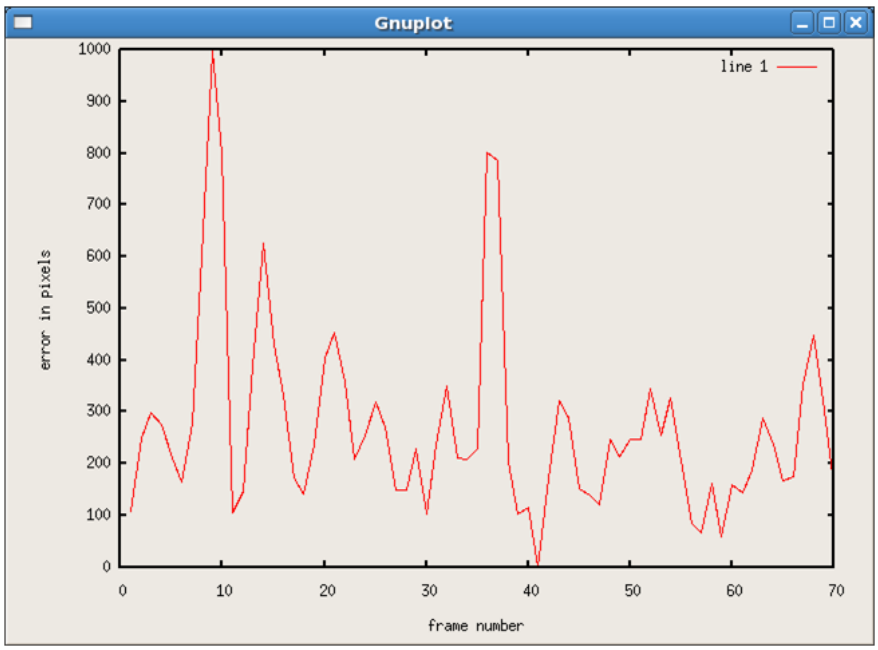


Figure 44. Total Alignment Error

6.5.2 Errors from IMU, Calibration in CLIF 2007

The error for an example CLIF 2007 camera 0 can be found in Figure 45 and Figure 46. In this case the average error is 103.95 pixels.

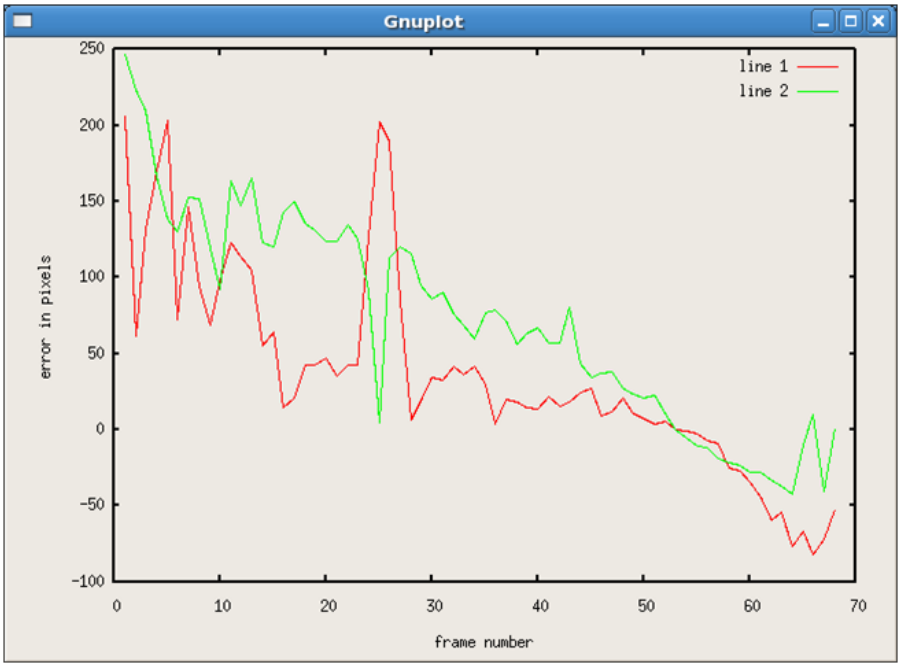


Figure 45. CLIF 2007 Example Alignment Error

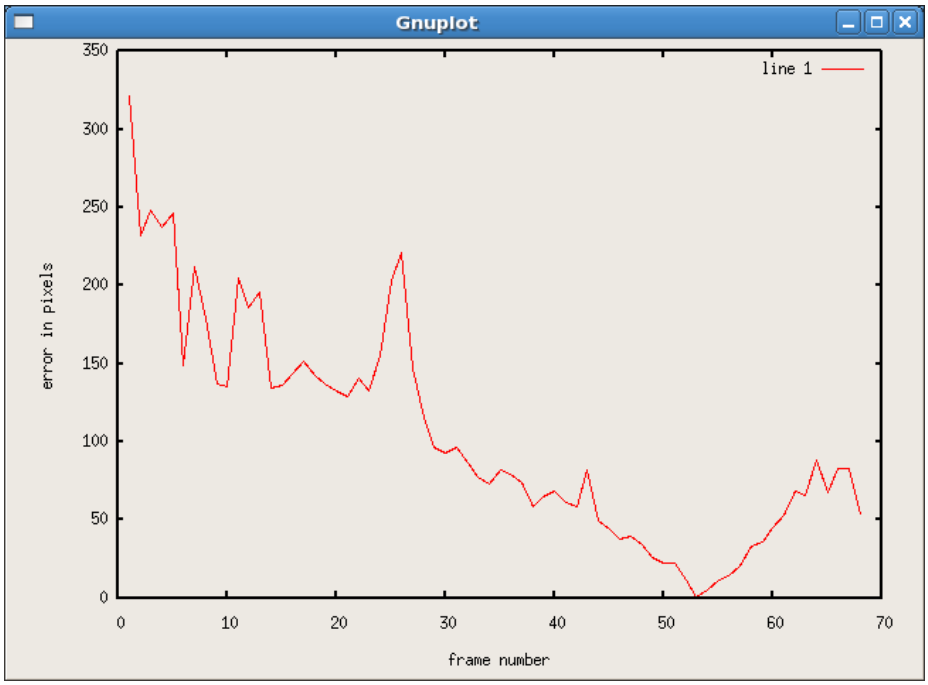


Figure 46. CLIF 2007 Absolute Alignment Error

6.5.3 Post SIFT Feature Stabilization Error

Figure 47 shows four frames from a stabilized example video included in this report. The left side of each frame is an extracted sub image of the scaled right side. The extracted sub image is the intersection of Woody Hayes Drive and Cannon Drive.

Table 6 shows the manually observed error at location (157, 89) of the video frames shown in Figure 47. The average error in X is -2.53 pixels, while the average error in Y is 2.07. At the scale of the georeferenced image, these errors correspond to -0.51 meters and 0.41 meters respectively. In the resulting example video included in this status report, it is intended that the zoomed-in left side of the frame show fine detail while the right side of the frame shows a scaled version of the entire stabilized frame.

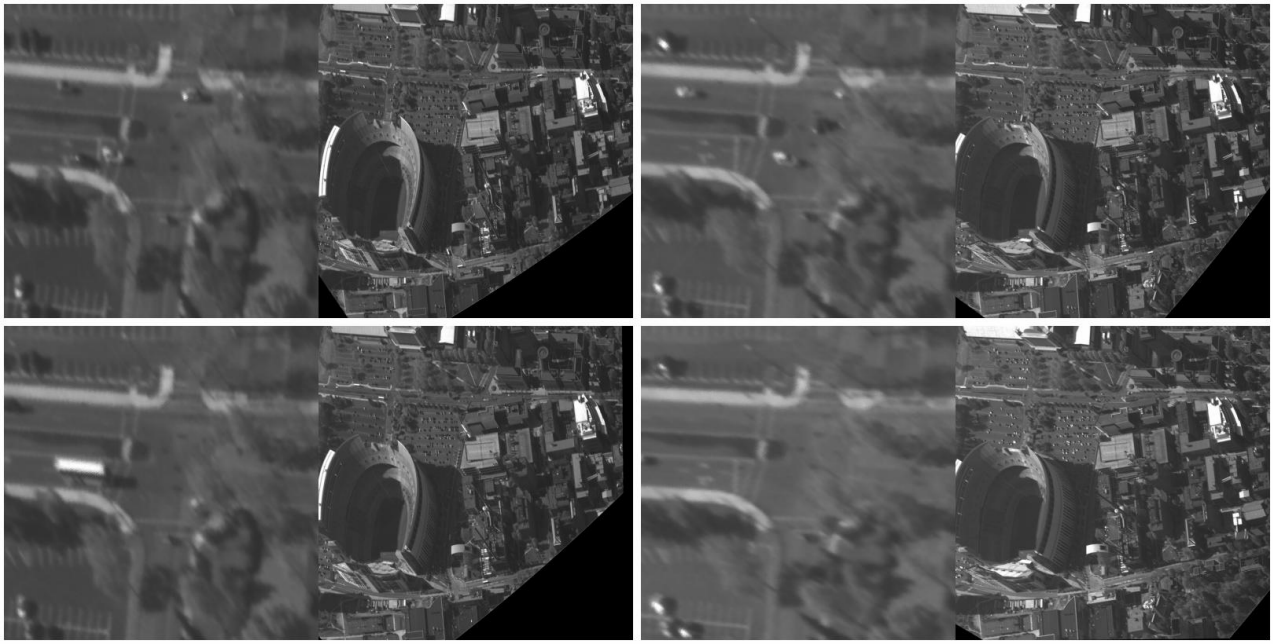


Figure 47. Four Frames from a Stabilized Video

Frame #	Original X	Original Y	Current X	Current Y	X Error	Y Error	X Error Meters	Y Error Meters
1	157	89	157	89	0	0	0	0
2	157	89	159	88	-2	1	-0.4	0.2
3	157	89	159	89	-2	0	-0.4	0.0
4	157	89	159	86	-2	3	-0.4	0.6
5	157	89	159	87	-2	2	-0.4	0.4
6	157	89	161	86	-4	3	-0.8	0.6
7	157	89	160	86	-3	3	-0.6	0.6
8	157	89	161	86	-4	3	-0.8	0.6
9	157	89	159	87	-2	2	-0.4	0.4
10	157	89	160	88	-3	1	-0.6	0.2
11	157	89	162	87	-5	2	-1.0	0.4
12	157	89	159	87	-2	2	-0.4	0.4
13	157	89	160	87	-3	2	-0.6	0.4
39	157	89	159	87	-2	2	-0.4	0.4
42	157	89	158	86	-1	3	-0.2	0.6
60	157	89	158	87	-1	2	-0.2	0.4
		Average Error			-2.53	2.07	-0.51	0.41

Table 6. Measured Error at Image Location (157, 89)

6.5.4 Problems Encountered with the CLIF2006 Data

For several reasons, the CLIF2006 dataset proved problematic. The images in the dataset were taken by a shorter focal length lens than the CLIF2007 data (approx 9000 pixels versus approx 15000 pixels), and the cameras were flown at a lower altitude (approx 7300ft versus approx 9200ft). Also, the viewing angle of the CLIF2006 dataset was very shallow. The flat viewing angle of the CLIF2006 data caused the georeferenced version of image 000003-003051 to

largely lie outside the 1.8 km by 1.5 km output region as shown in Figure 17. In contrast, the CLIF2007 data tended not to lie outside the 1.8 x 1.5 km output region at all. The flat viewing angle also causes the CLIF2006 data to have significant parallax effects when viewing objects which lie off the ground plane. As an expedient, the SIFT template frame is modified such that features from tall objects are masked away. However, the masking of tall structures cannot change the fact that the SIFT features are not holding up particularly well as the aircraft circles. The aggregate changes in geometry and lighting conditions mean that a single template frame's SIFT descriptors cannot continue to make correspondences for more than 100 frames or so.

A fix for this problem would be to abandon using a masked single template frame, and instead use a database of SIFT descriptors which are not tied to a single template at all.

6.6 Conclusion with Recommendations for Future Use of SIFT

The 2D affine transform cannot generally register aerial images. One method of dealing with this inability is to attempt to warp image points onto a 2D map by using the position and orientation of the camera. However, this method is fundamentally limited by the accuracy of the sensors which measure the camera's current position. An image to image registration step is still required to bring images into alignment.

Robust image to image registration is accomplished here by finding sets of image correspondences with the SIFT algorithm and using a combination of RANSAC and least squares to find a registering transform. In the two-image SIFT example above, images from the CLIF 2007 dataset which were 500 frames apart found 161 correspondences, revealing a substantial degree of robustness. The CLIF 2006 dataset was more problematic, but the CLIF2006 dataset viewing angle was quite shallow in comparison to the CLIF 2007 dataset. A shallow viewing angle combined with lower aircraft altitude and wider field of view mean that the viewing aspect of the ground changes significantly as the aircraft orbits. Improved use of the SIFT algorithm would incorporate SIFT descriptors from multiple template frames rather than the single template frame used on the CLIF2006 dataset.

If SIFT proves to be robust enough, it might be possible to create a database of SIFT descriptors. This database of SIFT descriptors could be annotated with longitude, latitude and elevation. The IMU's position and orientation information could be used to limit the SIFT descriptor database search to the SIFT descriptors that the camera “should” be seeing.

7. Conclusion

With the number of small and inexpensive UAVs increasing, it is feasible to build sensor networks for persistent sensing. In this report, we presented a new RDA algorithm for precision registration. Addressing some issues occurring in a persistent sensor network, we apply RDA for three problems: Multi-level/Multi-resolution data fusion, Multi-modal sensor fusion, and Mosaic construction.

The main result can be summarized as follows:

- Multi-scale and multi-resolution approach can be useful for both scale factor search and an optimal subset of large feature data.
- Information-theoretic RDA algorithm can help combine different modalities such as electro-optic and infrared sensor. Hence, we can register an additional and complementary information layer on top of the base information.
- Mosaic construction and scene localization are still challenging due to an error accumulation, even though our preliminary result can efficiently explain a scene with one image instead of 200 image frames. The trajectory of center images could indicate the looping closing problem for an autonomous navigator.

Future work should aim to achieve the following:

- Multi-modal video registration: it would be interesting to combine both the IR and the EO videos, and generate a new video sequence that has an additional and complementary layer of other information.
- A new metric for RDA and Mosaic quality: the average norm of local motion vectors near invariant features can be a good candidate.
- Geo-indexing or metadata fusion for both UAV navigators and layered video registration: we will investigate the methodology to couple the current data alignment scheme with the problem of UAV dynamics and its self-localization.
- Multi-modal fusion based persistent filtering: successful data fusion will lead to the overall performance improvement for tracking and surveillance by utilizing the current updated reference.

Appendix A
General Description of How to Use RDA Code

A.1 INTRODUCTION

This document explains how to build and use the klt_engine and RDA programs. The klt_engine implements the Kanade-Lucas-Tomasi Feature Tracker to find points of interest (mostly corners) in pgm image files and attempts to track those points through subsequent images. The output of this program can be passed to the RDA search program, which attempts to find an optimal transformation between two sets of feature points. The two programs can be used in tandem to analyze successive video frames to find the way the camera moved between frames.

The remainder of this Appendix A is broken into four sections, explained below:
QUICKSTART describes how to get up and running quickly using the klt_engine and RDA programs.
BUILD INSTRUCTIONS describes how to build the programs.
USE INSTRUCTIONS details common uses for the programs.
DETAILS/Additional Features shows miscellaneous details, including the input and output data formats for RDA and klt_engine, with three additional features.

A.2 QUICKSTART

There are two shell scripts in the main directory to quickly get you up and running with the contained programs. To first build the program, run:

```
$ ./build_me.sh
```

This script builds the programs with the most common optimization flag, and places the built programs (RDA and klt_engine) into the main directory. Next, use the perl script in the main folder to analyze successive frames, like this:

```
$ ./find_transforms.pl
```

This perl script reads the files klt_params and RDA_params for the parameters you want to run the klt_engine and RDA algorithm with. The files are initially loaded with the default values for each variable. For instance, if you run it as it is, you will get something like:

```
1.0021  0.00042714   -0.014434    0.99974    -1.9429   -0.10613;
1.0005  0.00042739   -0.019781    0.99963   -0.64177   -0.014042;
```

where the first line is the transformation between img0 and img1 and the second line is the transform between img1 and img2.

Examining klt_params and RDA_params shows you how to change the values. Additionally, you can comment out a line with a pound sign (#) at the beginning of the line to simply accept the default value. The only field that is required is -f [file list] for the klt_engine.

A.3 BUILD INSTRUCTIONS

To compile the program, type make at the command line. The makefiles used in building the program read in user supplied compiler flags in the environment variable BUILDFLAGS. As an example, if you want to compile with the flags -O2 and -Wall enabled, run:

```
$ make BUILDFLAGS="-O2 -Wall"
```

or, if you want to have the options persist between builds,

```
$ export BUILDFLAGS="-O2 -Wall"
$ make
```

We highly suggest passing the flag "-O3" to the compiler using the BUILDFLAGS variable. This optimization flag can greatly reduce the runtime of both the klt_engine and RDA binaries (depending on your system).

A.4 USE INSTRUCTIONS

klt_engine takes a list of pgm image files as command line arguments, finds "interesting" points in the images, and outputs those points as a list of feature points to standard out in a binary format. The image files are processed in the order they are passed to the klt_engine, and the feature points are output in the same order the image files are processed.

RDA reads a list of feature points from the standard input. RDA reads feature points in a binary format that matches the feature points generated by klt_engine. The output is printed on standard out, with one line per calculated transform. Therefore, if there are N sets of feature points input on standard in, and the command line specifies the number of
iterations to be M, RDA will print M*(N-1) lines of output, one per calculated transform.

A typical experiment may be to find the transforms between ten separate video frames. Assuming the video frames are stored as ten pgm image files, named img0.pgm through img9.pgm, the experiment can be run as a single line:

```
$ ./klt_engine -f img[0-9].pgm | ./RDA
```

The output for this will resemble:

```
   1.0021   0.0006443   -0.012911     1.0001    -2.1373    -0.20272;
   1.0005  0.00043043   -0.019785    0.99963    -0.63912   -0.014061;
   1.0004 -2.8869e-05    -0.01505          1    -1.4562    0.012806;
   1.0013   0.0017936   -0.014382     1.0002     -1.707     -0.4164;
  0.99904  0.00078372   -0.018393    0.99941    -0.51247   -0.056753;
    1.001  0.00053601   -0.012109    0.99955    -1.9443   -0.067654;
  0.99984   0.0011855   -0.014857    0.99847     -1.3059     0.01586;
   1.0001  0.00089846   -0.017233    0.99978    -0.98253    -0.16618;
   1.0008  0.00085932   -0.010806    0.99909     -2.0087   -0.057117;
```

A.5 Details and Additional Features

It is possible that RDA will give slightly different results on different machines, using different compiler flags, and other, similar changes. Because of the way RDA searches the optimization space, even very slight differences in floating point rounding, reordering of floating point operations, and differing random number generation can change to final answer. In order to iron out bugs however, the same binary on the same machine given the same input will always give the same answer (there is a constant seed to the random number generator).

The klt_engine output and RDA input data formats are identical, to make running them together simple. This format is illustrated below:

INT(n),FLOAT(p0.x),FLOAT(p0.y),INT(p0.weight),
FLOAT(p1.x),FLOAT(p1.y),INT(p1.weight), ... ,
FLOAT(p(n-1).x),FLOAT(p(n-1).y),INT(p(n-1).weight)

, where INT or FLOAT is the C data type, and what it represents is in parenthesis. So, FLOAT(p0.x) holds the x value of the zeroth feature point, FLOAT(p1.y) holds the y value of the first feature point, etc. The weight of the feature point represents how many previous frames that particular feature point was present in. INT(n) holds the number of feature points for this particular image. In summary, for each image file, klt_engine outputs an integer that holds the number of feature points, and two floating point numbers and an integer for each feature point.

A quick code snippet to read this binary input data format is below. The snippet reads the data from standard in, puts x values in the px array, y values in the py array, and the weights in the w array.

```
read_num = fread(&input_size, sizeof(int), 1, stdin);
if(read_num == 1){
  for(i=0;i<input_size;i++){
     read_num = fread(&(px[i]),sizeof(float),1,stdin);
     read_num = fread(&(py[i]),sizeof(float),1,stdin);
     read_num = fread(&(w[i]),sizeof(int),1,stdin);
  }
     }
```

Three new features have been considered:

- Consideration of feature "weight" in cost: This feature utilizes the klt feature detection algorithm to assign a weight to features that survive several consecutive video frames. The weight is then used to place more importance to features that appear to be "stationary", and less importance to more "ourlier-prone" features. This is controlled by the KLT_WEIGHTS flag in the RDA program. For information on how to set (and unset) the flag, run RDA -h. The default value is 0, so that feature point weights are not used.
- Reuse of the previous transform, if the previous transform's error is less than the threshold defined by TRANSFORM_REUSE_THRESHOLD: This feature allows RDA to reuse the previously found transformation for the starting point of the next search, if the previous error is below the user-definable TRANSFORM_REUSE_THRESHOLD. This feature will hopefully speed up a search, if the transforms from one frame to the next are largelt stable.

The motivation for using a threshold is that if the previous transform had an unusually large error, it will probably be better to start over from scratch. The default value for TRANSFORM_REUSE_THRESHOLD is -1, which effectively disables the reuse. For more information on how to set this runtime variable,

```
run RDA -h.
```

- Elimination of points from the search based on the weight found in the klt step. This is performed using two thresholds: ELIM_POINTS_THRESHOLD, and MIN_RATIO_POINTS. ElimPointsThreshold is the target threshold -- so 0.5 throw out points whose weight is less than half of the maximum weight (adjusted for the minumum weight). However, since this may throw out lots of potentially useful points, MIN_RATIO_POINTS defines a hard minimum on the number of points to include in calculating RDA. To always include all the points in the calculations, set MIN_RATIO_POINTS to a value >= 1.

Appendix B
Code ("stabilize_one_frame", "geo_ref") Use Example

B.1 Code "stabilize_one_frame"

The program "stabilize_one_frame" was developed on a Linux Red Hat system with the libraries and include files for OpenCV and Gnu Scientific Library (GSL) in standard locations. Although "stabilize_one_frame"'s compilation has not been tested on other Linux distributions, it should compile easily as long as OpenCV and GSL are installed.

Compiling "stabilize_one_frame" should be as simple as issuing the "make" command after moving the current working directory to the "geo_reference" directory. One should see the following:

```
[martinj@crl13 sift_stab]$ make
gcc -g -I. -I/usr/include/opencv -c sift.c
gcc -g -I. -I/usr/include/opencv -c kdtree.c
gcc -g -I. -I/usr/include/opencv -c minpq.c
gcc -g -I. -I/usr/include/opencv -c xform.c
gcc -g -I. -I/usr/include/opencv -c utils.c
gcc -g -I. -I/usr/include/opencv -c keypoints.c
gcc -g -I. -I/usr/include/opencv -c affine_warp.c
gcc -g -I. -I/usr/include/opencv -o stabilize_one_frame stabilize_one_frame.c sift.o
kdtree.o minpq.o xform.o utils.o keypoints.o affine_warp.o -L/usr/local/lib/ -lm -lcv -
lhighgui -lgsl -lgslcblas
[martinj@crl13 sift_stab]$
```

The SIFT code compiled here (sift.c, kdtree.c minpq.c utils.c) was written by Rob Hess (hess@eecs.oregonstate.edu).

"stabilize_one_frame" may be invoked as shown below:

```
[martinj@crl13 sift_stab]$ ./stabilize_one_frame template.pgm 000003-002195.pgm
got 42 sift correspondences
number of matches 37
found 37 ransac matches
low res:
[  1.02554950   0.03187709]    [ 12.74184134  -642.96184678]
[  0.02683933   0.99919101]
got 26 sift correspondences
got 45 sift correspondences
got 8 sift correspondences
no features found!
got 25 sift correspondences
got 21 sift correspondences
got 9 sift correspondences
got 8 sift correspondences
got 9 sift correspondences
no features found!
```

```
got 0 sift correspondences
151 total sift correspondences
number of matches 138
found 138 ransac matches
final_xform!
[  1.02962831   0.02848925]    [ 10.18459888  -612.51569460]
[  0.00866851   0.99544744]
[martinj@crl13 sift_stab]$
```

The “stabilize_one_frame” program works by finding an affine transform between smaller low-resolution versions of the argument images (specifically, in the above case, it's template.pgm and 000003-002195.pgm). The argument images' dimensions are divided by eight. The resulting low resolution transform then informs the code where to look for corresponding 700x700 pixel sub images. The sub image correspondences may then be used to find high resolution SIFT correspondences. The locations of the sub images are specific to the template in the example, and they are selected such that they provide good cover for distinctive high contrast portions of the template. If another template is used, another set of sub image locations should be inserted into the code at line 66 which currently looks like:

```
//feature rich locations on the template
num_template_centers=11;
tc[0].x=753;  tc[0].y=891;
tc[1].x=1540;  tc[1].y=1452;
tc[2].x=3509;  tc[2].y=555;
tc[3].x=2700;  tc[3].y=2942;
tc[4].x=1144;  tc[4].y=3129;
tc[5].x=253;  tc[5].y=2717;
tc[6].x=3162;  tc[6].y=709;
tc[7].x=1650;  tc[7].y=720;
tc[8].x=3558;  tc[8].y=682;
tc[9].x=2689;  tc[9].y=176;
tc[10].x=1653; tc[10].y=2088;
```

where the above code is merely defining a list of sub image locations to look for high resolution SIFT correspondences.

B.2 Code “geo_ref”

The program “geo_ref” was developed on a Linux Red Hat system with the libraries and include files for OpenCV installed in standard locations. Although “geo_ref's” compilation has not been tested on other Linux distributions, it should compile easily as long as OpenCV is installed.

Compiling “geo_ref” should be as simple as issuing the “make” command after moving the current working directory to the “geo_reference” directory. One should see the following:

```
[martinj@crl13 geo_reference]$ make
gcc  -I. -c xform.c
gcc  -I. -c xform_io.c
gcc  -DCAMERA3_2006 -I. -o geo_ref geo_ref.c xform.o xform_io.o -lm -lcv
[martinj@crl13 geo_reference]$
```

The "make" command invokes the Gnu C compiler (gcc) and compiles the code into the "geo_ref" executable. In this case, the "-DCAMERA3_2006" controls compilation such that the parameters for camera 3 from the CLIF2006 dataset are compiled into the code. It is possible to modify the Makefile and substitute "-DCAMERA0_2007" or "-DCAMERA1_2007" which will conditionally compile "geo_ref" for camera 0 CLIF2007 or camera 1 CLIF2007 rather than camera 3 CLIF2006. If the user attempts to georeference camera data for which the code is not currently compiled, the resulting image will be in a strange spot, potentially off the output screen.

Example camera data from camera 0 CLIF2007, camera 1 CLIF2007 and camera 3 CLIF2006 are included. Invoking "geo_ref" to georeference camera 3 CLIF2007 should look like the following:

```
[martinj@crl13 geo_reference]$ ./geo_ref ./cam3_2006_example/000003-002094.pgm
./cam3_2006_example/000000-002094.txt
using camera 3 2006 parameters
image size 2672 4008
output image size 9008 7672 at 0.200000 meters per pixel
[martinj@crl13 geo_reference]$
```

The two input files included as parameters to the "geo_ref" executable are a .pgm image file containing the raw image and a .txt position and orientation file.

"geo_ref" will write the output file in the current working directory as "out.pgm". In this particular case, the output image in Figure B1 is the georeferenced output based on the position and orientation of the camera. This image is 1.8 x 1.5 km and centered on the stadium.

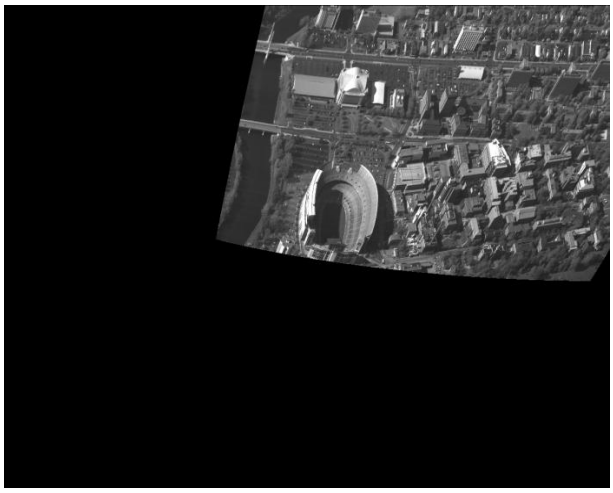


Figure B1

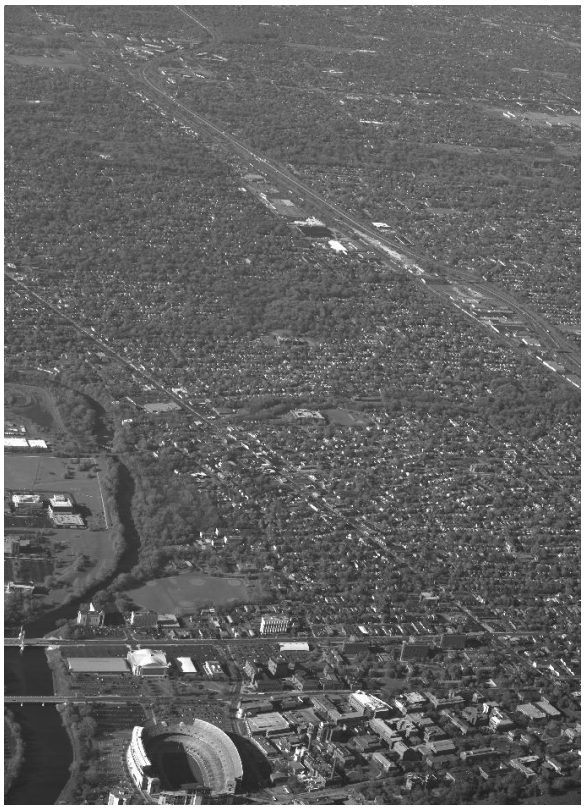


Figure B2

The input file for the image in Figure B1 is shown in Figure B2. One can see that most of the input image has been cropped from the output since most of the image is a long distance away from the center location at the stadium.

Similar output files may be obtained for the camera 0 CLIF2007 example, which is shown in Figures B3. This image also represents a region 1.8 x 1.5 km centered on the stadium. However, the CLIF2007 image was taken with a longer focal length lens which was pointed more directly downward. Thus, the output image is contained in the 1.8x1.5 km region about the stadium.

Figure B3

B.3 Code "descend"

The program "descend" was developed on a Linux Red Hat system. You can also find the libraries and include files for OpenCV. Although its compilation has not been tested on other Linux distributions, it should compile easily as long as OpenCV.

Compiling "descend" should be as simple as issuing the "make" command after moving the current working directory to the "descend" directory. One should see the following:

```
[martinj@crl13 descend]$ make
gcc  -g -I. -c xform.c
gcc  -g -I. -c xform_io.c
gcc -g -I. -o descend descend.c xform.o xform_io.o -lm -lcv
[martinj@crl13 descend]$
```

"descend" may be invoked as shown below:

```
[martinj@crl13 descend]$ ./descend 000003-003051.pgm 000000-003051.txt corrs_2006_3051
```

"descend" is highly experimental program which is tweaked and modified by making changes to C code and recompiling. The correctness of the code is shown only by the anecdotal evidence that the error decreases with iterations.

The specific example included here performs the relatively simple action of finding the camera orientation offset. The first output file, "start.ppm", is the alignment provided by the initial conditions. In the example provided here, this image may be found in Figure B4. The second output file, "end.ppm", shows the alignment which resulted from the gradient descent. The image found in "end.ppm" may be found in Figure B5.

The initial conditions for the alignment shown in Figure B4 are shown in the following table.

orientation offset angle in radians	0.0
orientation offset axis x component	1.0
orientation offset axis y component	0.0
orientation offset axis z component	0.0
focal length x component in pixels	8838.48
focal length y component in pixels	9488.43
optical image center x component in pixels	2005.34
optical image center y component in pixels	1299.66
lens distortion coefficient k1	0.091610
lens distortion coefficient k2	0.407229
lens distortion coefficient k3	0.023974
lens distortion coefficient p1	0.007713
lens distortion coefficient p2	0.011477

Note in the above table that there is no orientation offset. The (wrong) assumption is that CLIF2007 camera 0 is aligned with the camera head.

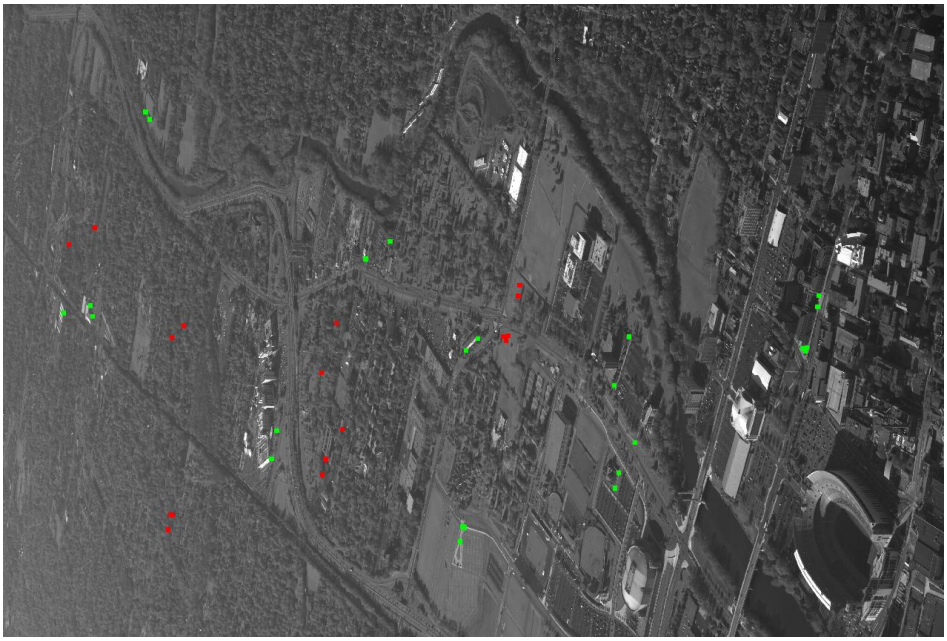


Figure B4

In Figure B4, the red and green dots are far from being aligned. However, the pattern of dots is similar and one could imagine alignment occurring if all the red dots were shoved about a third of an image width to the right. The "descend" program will modify the initial conditions such that this occurs. The table below shows the results of invoking the "descend" program as described above.

orientation offset angle in radians	-0.140906
orientation offset axis x component	0.998671
orientation offset axis y component	0.039754
orientation offset axis z component	0.032792
focal length x component in pixels	8838.48
focal length y component in pixels	9488.43
optical image center x component in pixels	2005.34
optical image center y component in pixels	1299.66
lens distortion coefficient k1	0.091610
lens distortion coefficient k2	0.407229
lens distortion coefficient k3	0.023974
lens distortion coefficient p1	0.007713
lens distortion coefficient p2	0.011477

The values in the above table are identical except for the orientation offset parameters. The orientation offset is about -8 degrees about the x-axis (roll) axis.

The above values resulted in the alignment shown in Figure B5, which is better than the initial alignment shown in Figure B4.

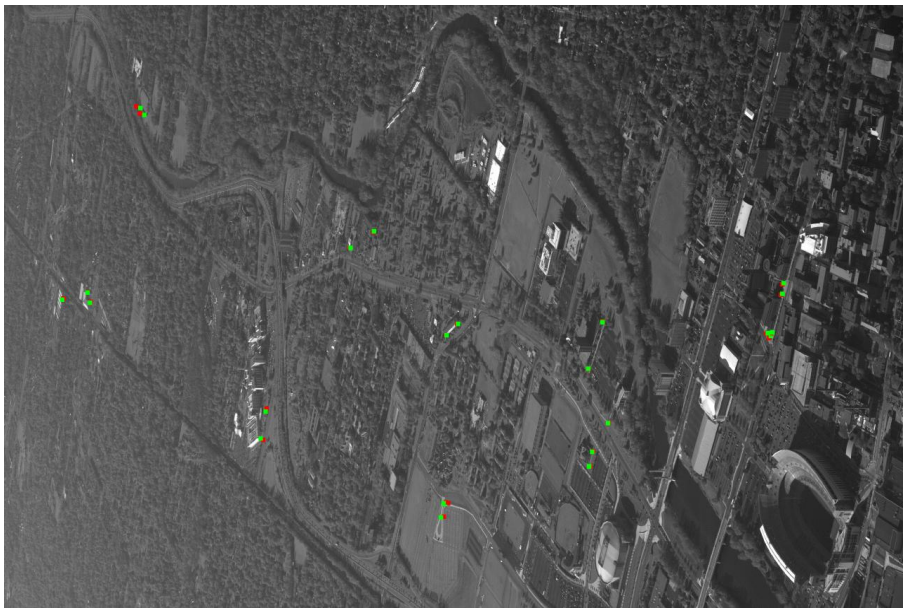


Figure B5

Camera parameters other than the orientation offset did not change in the gradient descent because in this example the only parameters allowed to change were the orientation parameters. This parameter selection can be accomplished by modifying the following assignments in "descent.c":

```
//gain for descent step
#if 1
  descent_scale.angle=-0.00001;
  descent_scale.x=-0.0001;  descent_scale.y=-0.0001;  descent_scale.z=-0.0001;
  descent_scale.fc_x=0; descent_scale.fc_y=0;
  descent_scale.cc_x=0;  descent_scale.cc_y=0;
  descent_scale.k[0]=0;  descent_scale.k[1]=0;  descent_scale.k[2]=0;
descent_scale.k[3]=0;
  descent_scale.k[ 4]=0;
#endif
```

Note that the only fields which are not zero are the camera orientation offset fields. It is fairly common to "turn off" certain descent directions. Often, the parameters may accomplish similar things, and the user probably does not want to fix an orientation misalignment by descending into strange lens distortion coefficients. Thus, optimizing on only some of the coefficients is common.

The initial conditions may be changed by modifying the following "descent.c" code:

```
 //  cd.angle=-0.141219;
  cd.angle=0.0; //start at zero instead of correct value above
  // cd.x=0.998639; cd.y=0.041411; cd.z=0.031710;
  cd.x=1.0; cd.y=0.0; cd.z=0.0; //start with x axis instead of correct value above

  cd.fc_x=8838.480381; cd.fc_y= 9488.430025;
  cd.cc_x= 2005.340061;  cd.cc_y= 1299.661670;
```

In order to create this example, I modified the initial conditions such that there was no camera orientation offset. Thus, the one can see the original values commented out.

8. References

[1] **Jwa, S., Özgüner, Ü. and Tang, Z.** Information-theoretic Data Registration for UAV-based sensing. *IEEE Trans. on Intelligent Transportation Systems.* March 2008, Vol. 9, No. 1, pp. 5-15.

[2] **Jwa, S., Tang, Z. and Özgüner, Ü.** Robust Data Alignment Based on Information Theory and Its Applications in Road Following Situation. *IEEE International conference on Intelligent Transportation Systems.* September 17-20, 2006, pp. 1328-1333.

[3] **Jwa, S. and Özgüner, Ü.** Multi-UAV Sensing Over Urban Areas Via Layered Data Fusion. *IEEE Statistical Signal Processing Workshop.* August 26-29, 2007, pp. 576-580.

[4] —. Problems in Data Registration for Persistent Sensing. *Intelligent Computing: Theory and Applications VI, SPIE Defense & Security* . March 16-20, 2008, Vol. 6961.

[5] **Modersitzki, J.** *Numerical Methods for Image Registration.* London, U. K. : Oxford University Press, 2004.

[6] **Zitova, B. and Flusser, J.** Image registration methods: a survey. *Image and Vision Computing.* October 2003, Vol. 21, No. 11, pp. 977-1000.

[7] **Luersen, M. A. and Le Riche, R.** Globalized Nelder-Mead method for engineering optimization. *Computers & Structures.* September-October 2004, Vol. 82, Issues 23-26, pp. 2251-2260.

[8] **Wang, Y., Wood, K. and McClain, M.** Information-Theoretic Matching of Two Point Sets. *IEEE Trans. on Image Processing.* August 2002, Vol. 11, No. 8, pp. 868-872.

[9] **Cover, T. M. and Thomas, J. A.** *Elements of Information Theory.* New York : Wiley, 1991.

[10] **Duda, R. O. and Hart, P. E.** *Pattern Classification and Scene Analysis.* Hoboken, NJ : Wiley, 1973.

[11] **Akaike, H.** Information Theory and an Extension of the Maximum Likelihood Principle. *2nd International Symposium on Information Theory.* 1973, pp. 267-281.

[12] **Lagarias, J. C., et al.** Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization.* May 1998, Vol. 9, No. 1, pp. 112-147.

[13] **Spall, J. C.** *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control.* Hoboken, NJ : Wiley, 2003.

[14] **Lucas, B. D. and Kanade, T.** An iterative image registration technique with an application to stereo vision. *Proceedings of the International Joint Conference on Artificial Intelligence.* April 1981, pp. 674-679.

[15] **Hartley, R. and Zisserman, A.** *Multiple View Geometry in Computer Vision.* Cambridge, U. K. : Cambridge University Press, 2004.

[16] **Fischler, M. A. and Bolles, R. C.** Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM.* June 1981, Vol. 24, No. 6, pp. 381-395.

[17] **Kumar,R., Sawhney, H., Samarasekera, S., Hsu, S., Tao, H., Guo, Y., Hanna, K., Pope, A., Wildes, R., Hirvonen, D., Hansen, M., Burt, P.** Aerial Video Surveillance and Exploitation. *Proceedings of the IEEE, Special Issue on Third Generation Surveillance Systems.* October 2001, Vol. 89, No. 10, pp. 1518-1539.

[18] **Angel, A., Hickman, M., Mirchandani, P. and Chandnani, D.** Methods of Analyzing Traffic Imagery Collected From Aerial Platforms. *IEEE Trans. on Intelligent Transportation Systems.* June 2003, Vol. 4, No. 2, pp. 99-107.

[19] **Schowengerdt, A. C. Shastry and R. A.** Airborne Video Registration and Traffic-Flow Parameter Estimation. *IEEE Trans. on Intelligent Transportation Systems.* December 2005, Vol. 6, No. 4, pp. 391-405.

[20] **Xiao, J. and Shah, M.** Two-Frame Wide Baseline Matching. *International Conference on Computer Vision* . 2003, Vol. 2, pp. 603-609.

[21] **Dufournaud, Y., Schmid, C. and Horaud, R.** Image matching with scale adjustment. *Computer Vision and Image Understanding.* 2004, Vol. 93, pp. 175-194.

[22] **Shah, M. and Kumar, R. (Eds,).** *Video Resigstration.* Boston, MA : Kluwer Academic Publishers, 2003.

[23] **Szeliski, R.** Video Mosaics for Virtual Environments. *IEEE Computer Graphics and Applications.* March 1996, Vol. 16, No. 2, pp. 21-30.

[24] **Lindeberg, T.** *Scale-Space Theory in Computer Vision.* Boston, MA : Kluwer, 1994.

[25] **Sporring, J. and Weickert, J.** Information Measures in Scale-Spaces. *IEEE Trans. on Information Theory.* April 1999, Vol. 45, No. 3, pp. 1051-1058.

[26] **Jahne, B.** *Digital Image Processing.* London, U. K. : Springer, 2005.

[27] **Lowe, D. G.** Object recognition from local scale-invariant features. *Proceedings of International Conference on Computer Vision.* September 1999, pp. 1150-1157.

[28] **Gonzalez, M. G., Holifield, P. and Varley, M.** Improved Video Mosaic Construction by Accumulated Alignment Error Distribution. *Proceedings of British Machine Vision Conference.* September 1998, pp. 377-387.

[29] **Williams, D.** *Probability with Martingales.* Cambridge, U. K. : Cambridge University Press, 1991.

[30] **Chen, H., Varshney, P. K. and Arora, M. K.** Performance of Mutual Information Similarity Measure for Registration of Multitemporal Remote Sensing Images. *IEEE Trans. on Geoscience and Remote Sensing.* November 2003, Vol. 41, No. 11, pp. 2445-2454.

[31] **Cole-Rhodes, A. A., Johnson, K. L., LeMoigne, J. and Zavorin, I.** Multiresolution Registration of Remote Sensing Imagery by Optimization of Mutual Information Using a Stochastic Gradient. *IEEE Trans. on Image Processing.* December 2003, Vol. 12, No. 12, pp. 1495-1511.

[32] **Govindu, V. and Shekhar, C.** Alignment using distributions of local geometric proporties. *IEEE Trans. Pattern Anal. Machine Intell.* October 1999, Vol. 21, No. 10, pp. 1031-1043.

[33] **Pluim, J. P. W., Maintz, J. B. A. and Viergever, M. A.** Mutual information based registration of medical images: a survey. *IEEE Trans. on Medical Imaging.* August 2003, Vol. 22, No. 8, pp. 986-1004.

[34] **Madhavan, R., Hong, T. and Messina, E.** Temporal Range Registration for Unmanned Ground and Aerial Vehicles. *Proceedings of the IEEE International Conference on Robotics and Automation.* April 2004, pp. 3180-3187.

[35] **Viola, P. and Wells, W.** Alignment by Maximization of Mutual Information. *Proceedings of 5th International Conference on Computer Vision.* 1995, pp. 16-23.

[36] **Viola, P.** *Alignment by Maximization of Mutual Information.* Department Electrical Engineering and Computer Science. Cambridge, MA : MIT, 1995. PhD Dissertation.

LIST OF ACRONYMS, ABBREVIATIONS, AND SYMBOLS

ACRONYM	DESCRIPTION
BSP	Binary Space Partitioning
CITR	Control and Intelligent Transportation Research
CLIF	Columbus Large Image Format
DEM	Digital Elevation Models
DOG	Difference Of Gaussians
EO	Electro-optical
ERDA	Extended Robust Data Alignment
IMU	Inertial Measurement Unit
IR	Infrared
KLT	Kanade-Lucas-Tomasi
LS	least squares
RANSAC	Random Sample Consensus
RASER	Revolutionary Automatic Target Recognition and Sensor Research
RDA	Robust Data Alignment
SIFT	Scale Invariant Feature Transform
UAV	Unmanned Aerial Vehicles
VIVID	Video Verification of Identity